Semantics of
Multiple Inheritance with Exceptions
in Hierarchically Structured
Logic Theories

Gabriel de Sousa Torcato David.
Author: Gabriel David
Title: Semantics of Multiple Inheritance with Exceptions in Hierarchically Structured Logic Theories (PhD thesis)
Supervisor: António Porto
Institution: Universidade Nova de Lisboa
           Departamento de Informática
Address: Quinta da Torre
        P-2825 Monte da Caparica
        Portugal
Copyright: Universidade Nova de Lisboa
Place: Lisboa
Date: 1994
Abstract

The thesis studies the composition of theories expressed by extended logic programs arranged in a hierarchy. The scope is restricted to the propositional case without default negation but encompasses multiple inheritance and exceptions. Exceptions are expressed through the combination of contradiction and specificity. The hierarchy is strict but rules, understood as conditional properties, are defeasible. Ambiguities are resolved by neutralization and exceptions give rise to overriding.

Most of the problems addressed are common to those raised by the works on inheritance networks. These, in conjunction with the efforts towards the structuring of logic programs, were the main motivation of this work. The methodology followed borrows several techniques used in the semantics of Logic Programming.

The language presented deals with both positive and negative information in a symmetric way and it is capable of expressing conjunction. Its meaning is defined by an extensional semantics which relies on the notion of typical individuals. It is parametric on the specification of the inheritance method. The combination of two simple principles, those of a static hierarchy and of conditional properties, results in a dynamic notion of specificity. It is described by the interplay of two mutually defined relations. One represents the skeptical semantics and the other subsumes the credulous alternatives. They are stated in terms of the least fixed-point of a bottom-up operator. The operational semantics is studied in conjunction with a translation to Logic Programming and proved to be sound and complete with respect to the bottom-up semantics, if loop detection is used. It is still shown to be equivalent to a suitable stability operator. An embedding in the argumentation theory highlights the nonmonotonic behavior of the system.

Keywords: artificial intelligence, automated reasoning, nonmonotonic reasoning, structured logic programming, multiple inheritance with exceptions.
Resumo

A tese estuda a composição de teorias expressas por programas em lógica estendidos e organizadas numa hierarquia. O âmbito é restrito ao caso proposicional sem negação por omissão, mas abrange herança múltipla e exceções. As exceções são expressas pela combinação da contradição com a especificidade. A hierarquia é estrita mas as regras, entendidas como propriedades condicionais, são canceláveis. As ambiguidades são resolvidas por neutralização e as exceções causam revogação ("overriding").

A maior parte dos problemas encarados é comum aos levantados pelos estudos em redes de herança. Estes, em conjunto com os esforços no sentido da estruturação de programas em lógica, foram a principal motivação deste trabalho. A metodologia seguida socorre-se de várias técnicas utilizadas na semântica da Programação em Lógica.

A linguagem apresentada trata a informação quer positiva quer negativa de forma simétrica e é capaz de exprimir a conjunção. O seu significado é definido por uma semântica extensional que assenta na noção de indivíduos típicos. Esta é paramétrica na especificação do método de herança. A combinação de dois princípios simples, os de hierarquia estática e de propriedades condicionais, resulta numa noção dinâmica de especificidade. Esta é descrita pela interacção de duas relações mutuamente definidas. Uma representa a semântica céptica e a outra subsume as alternativas crêdulas. Ambas são definidas em termos do menor ponto fixo de um operador ascendente. A semântica operacional é estudada juntamente com uma tradução para Programação em Lógica e prova-se ser coerente e completa relativamente à semântica ascendente se se usar detecção de ciclos. Mostra-se ainda que ela é equivalente a um operador de estabilidade apropriado. Uma formalização na Teoria da Argumentação realça o comportamento não monotónico do sistema.

Descritores: inteligência artificial, raciocínio automático, raciocínio não monotónico, programação em lógica estruturada, herança múltipla com exceções.
Analyse

La thèse étudie la composition de théories exprimées en programmation logique étendue et organisées selon une hiérarchie. Le sujet est restreint au cas propositionnel sans négation par omission mais il comprend héritage multiple et exceptions. Les exceptions sont exprimées par une combinaison de contradiction et de spécificité. La hiérarchie est stricte mais les règles, interprétées comme des propriétés conditionnelles, sont annulables. Les ambiguités sont résolues par neutralisation et les exceptions originent la révocation ("overriding").

La plupart des problèmes envisagés est commune aux problèmes abordés par les études sur les réseaux d'héritage. Ceux-ci, de même que les efforts vers la structuration des programmes logiques, ont été la principale motivation de ce travail. La méthodologie suivie emprunte plusieurs techniques utilisées pour la sémantique de la Programmation Logique.

Le langage présenté traite l'information positive ainsi que la négative d'une façon symétrique et est capable d'exprimer la conjonction. Sa signification est définie par une sémantique extensionnelle qui s'appuie sur la notion d'individus typiques. Celle-ci est paramétrisée par la spécification de la méthode d'héritage. La combinaison de deux principes simples, une hiérarchie statique et des propriétés conditionnelles, aboutit à une notion dynamique de spécificité, laquelle est décrite par l'interaction de deux relations mutuellement définies. L'une représente la sémantique sceptique et l'autre est un surensemble des alternatives crédibles. Elles sont spécifiées par le moindre point fixe d'un opérateur ascendant. La sémantique opérationnelle est étudiée avec une traduction en Programmation Logique et on prouve qu'elle est cohérente et complète par référence à la sémantique ascendant, si l'on emploie la détection de cycles. On montre encore qu'elle est équivalente à un opérateur de stabilité approprié. Une formulation selon la Théorie de l'Argumentation éclaire le comportement non-monotone du système.

Mots-clés: intelligence artificielle, raisonnement automatique, raisonnement non-monotone, programmation logique structuré, héritage multiple avec exceptions.
Acknowledgments

First of all, I want to thank my parents for contributing during so many years to what I am. I apologize to my children and wife for all the time I have stolen from them.

I thank my supervisor, António Porto, for so many hints and discussions on the thesis and a lot of other subjects and for teaching me how to do scientific research, with his patience and exigency. I am also grateful to Luís Monteiro for some clarifying discussions and technical help on the hardest points. To both I express my gratitude for receiving me in their office and for the warm and stimulating personal environment.

I acknowledge the support of the following institutions:

Departamento de Informática UNL for receiving me and providing a good working environment, both in terms of material conditions and of opportunities to meet international level researchers.

INESC for a one-year long grant and for lending a computer, and specially Borges Gouveia who created the opportunity and incited me to come to the UNL.

Faculdade de Engenharia UP for making my life as a lecturer easy.

INIC for a four-year grant.

JNICT, ESPRIT JNICT project PEROLA and ESPRIT projects ALPES and BRA 3020 “Integration” partially supported my work.

I am also willing to record my gratitude to all the colleagues at the UNL who directly or indirectly helped me in several ways, namely Cristina Ribeiro, Irene Rodrigues and Salvador Abreu, Vasco Pedro and Margarida Mamede, Luis Caires, Anabela Ribeiro, José Júlio Alferes, Paulo Rosado, Filipa Reis and many others.

To conclude, I thank Manuel Marques and Manuela Alves for extensive support and invaluable friendship.

I am aware of the impossibility of compensating everyone whom I am indebted to. My way of paying back will be to try to help the next generation the way they helped me.
Contents

0 Introduction .................................................. 1
  0.1 Subject delimitation ................................. 1
  0.2 Inheritance networks .............................. 4
  0.3 Original contributions of the thesis ............. 8
  0.4 Outline of the thesis .............................. 9

1 The framework: syntax and extensional semantics .... 11
  1.1 Hierarchically structured theories .............. 11
      1.1.1 Presentation .................................. 11
      1.1.2 The language of HST ...................... 14
  1.2 Extensional semantics ....................... 16
      1.2.1 Interpretations .............................. 16
      1.2.2 Models .................................. 19
      1.2.3 Standard models ........................... 23
      1.2.4 Entailment .................................. 26
      1.2.5 Models revisited ............................ 30
      1.2.6 Extending the language .................... 33
          1.2.6.1 Constructive semantics .............. 33
          1.2.6.2 Three-valued approach .............. 39
          1.2.6.3 About the rules .................... 41
      1.2.7 Complete information approaches .......... 42

2 Bottom-up construction of the characterization .... 47
  2.1 Predicate inheritance .............................. 47
      2.1.1 Models .................................. 51
      2.1.2 The basic overriding situation .......... 54
      2.1.3 Contradictory images .................... 60
  2.2 Rule inheritance ............................... 65
      2.2.1 Specificity .............................. 68
      2.2.2 Fixed-point definition of the characterization .... 73
      2.2.3 Behavior of the HST ...................... 81
          2.2.3.1 Restricted monotonicity .......... 81
          2.2.3.2 Dealing with ambiguity .......... 83
  2.3 Evaluation of the inheritance mechanisms ....... 93
3 Top-down inference .......... 95
  3.1 The inference system ......... 95
    3.1.1 Derivation relations ..... 95
    3.1.2 Derivations ............. 97
    3.1.3 Cyclic derivations ..... 101
  3.2 Translation to Logic Programming .......... 103
    3.2.1 Direct translation ..... 103
    3.2.2 Transformational approach ..... 104
  3.3 Proving the consistency ........ 109
    3.3.1 In the translation ..... 109
    3.3.2 From well-founded to stable semantics ..... 113
    3.3.3 In HST .......... 116
  3.4 Correctness of the inference system .......... 122
    3.4.1 Rank of a derivation ..... 122
    3.4.2 Completeness .......... 125
    3.4.3 Soundness ........ 128
  3.5 Reducing to finite failure .......... 130
    3.5.1 Explicitly modeling finite failure .......... 132
    3.5.2 Relative extensions .......... 136

4 Alternative views .......... 139
  4.1 Stable models of a class .......... 139
    4.1.1 Interpretations and models .......... 139
    4.1.2 Stability operator .......... 142
  4.2 HST as an Argumentation Theory .......... 151
    4.2.1 Introduction to the Argumentation Theory .......... 151
    4.2.2 Embedding HST in the Argumentation Framework .......... 155
      4.2.2.1 Arguments .......... 156
      4.2.2.2 Relation attacks .......... 158
      4.2.2.3 Correspondence .......... 160
    4.2.3 Generalizing the embedding .......... 162

5 Related work .......... 171
  5.1 HST versus inheritance networks .......... 171
    5.1.1 Translation .......... 171
    5.1.2 Ambiguity propagating and ambiguity blocking .......... 173
    5.1.3 A deeper reason for divergence .......... 175
    5.1.4 Expressiveness .......... 179
  5.2 Other logic approaches related to HST .......... 180

6 Conclusions .......... 183
  6.1 Prominent results .......... 183
  6.2 Possible developments .......... 186

A Meta-interpreter .......... 195
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Clyde, the elephant</td>
<td>5</td>
</tr>
<tr>
<td>0.2</td>
<td>Nixon double diamond</td>
<td>7</td>
</tr>
<tr>
<td>1.1</td>
<td>Two representations of the <em>bat</em> world.</td>
<td>12</td>
</tr>
<tr>
<td>1.2</td>
<td>Two americans.</td>
<td>20</td>
</tr>
<tr>
<td>1.3</td>
<td>Four interpretations for the HST <em>two americans</em>.</td>
<td>20</td>
</tr>
<tr>
<td>1.4</td>
<td>From classes to their union.</td>
<td>31</td>
</tr>
<tr>
<td>1.5</td>
<td>From classes to their intersection?</td>
<td>32</td>
</tr>
<tr>
<td>1.6</td>
<td>Inconsistent pre-interpretations.</td>
<td>33</td>
</tr>
<tr>
<td>1.7</td>
<td>Constructive disjunction.</td>
<td>38</td>
</tr>
<tr>
<td>1.8</td>
<td>Two-valued standard model.</td>
<td>44</td>
</tr>
<tr>
<td>2.1</td>
<td>Whale</td>
<td>50</td>
</tr>
<tr>
<td>2.2</td>
<td>Lattice of split scenarios.</td>
<td>55</td>
</tr>
<tr>
<td>2.3</td>
<td>A lower scenario that is not a classical model.</td>
<td>57</td>
</tr>
<tr>
<td>2.4</td>
<td>The odd and the even loops.</td>
<td>58</td>
</tr>
<tr>
<td>2.5</td>
<td>Meet versus intersection.</td>
<td>59</td>
</tr>
<tr>
<td>2.6</td>
<td>A class with its own facts.</td>
<td>60</td>
</tr>
<tr>
<td>2.7</td>
<td>A lattice for three defaults.</td>
<td>61</td>
</tr>
<tr>
<td>2.8</td>
<td>Contradiction in the consequences.</td>
<td>61</td>
</tr>
<tr>
<td>2.9</td>
<td>Indirect contradiction.</td>
<td>62</td>
</tr>
<tr>
<td>2.10</td>
<td>Exception of exception.</td>
<td>66</td>
</tr>
<tr>
<td>2.11</td>
<td>Postponed overriding.</td>
<td>67</td>
</tr>
<tr>
<td>2.12</td>
<td>Nixon dilemma.</td>
<td>68</td>
</tr>
<tr>
<td>2.13</td>
<td>Clyde, the elephant</td>
<td>70</td>
</tr>
<tr>
<td>2.14</td>
<td>A convoluted example.</td>
<td>71</td>
</tr>
<tr>
<td>2.15</td>
<td>A local neutralization.</td>
<td>71</td>
</tr>
<tr>
<td>2.16</td>
<td>The swimming cetaceous.</td>
<td>76</td>
</tr>
<tr>
<td>2.17</td>
<td>The division of a theory.</td>
<td>77</td>
</tr>
<tr>
<td>2.18</td>
<td>Overriding on top of neutralization.</td>
<td>87</td>
</tr>
<tr>
<td>2.19</td>
<td>Floating conclusion.</td>
<td>89</td>
</tr>
<tr>
<td>2.20</td>
<td>Cascaded neutralization.</td>
<td>91</td>
</tr>
<tr>
<td>2.21</td>
<td>Impossible conclusion.</td>
<td>92</td>
</tr>
<tr>
<td>Section</td>
<td>Figure Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>3.1</td>
<td>Inference rules.</td>
<td>96</td>
</tr>
<tr>
<td>3.2</td>
<td>A derivation using the cascaded neutralization.</td>
<td>99</td>
</tr>
<tr>
<td>3.3</td>
<td>Auto-overflowing.</td>
<td>101</td>
</tr>
<tr>
<td>3.4</td>
<td>Two markings.</td>
<td>101</td>
</tr>
<tr>
<td>3.5</td>
<td>Simplified version of the meta-interpreter.</td>
<td>104</td>
</tr>
<tr>
<td>3.6</td>
<td>The swimming cetaceous, again.</td>
<td>105</td>
</tr>
<tr>
<td>3.7</td>
<td>Well-founded semantics of the program in step 2.</td>
<td>108</td>
</tr>
<tr>
<td>3.8</td>
<td>Almost bi-stable.</td>
<td>114</td>
</tr>
<tr>
<td>3.9</td>
<td>Well-founded model with undefined symbols.</td>
<td>114</td>
</tr>
<tr>
<td>3.10</td>
<td>Stable extensions of the well-founded model.</td>
<td>115</td>
</tr>
<tr>
<td>3.11</td>
<td>Translation with weak overriding.</td>
<td>116</td>
</tr>
<tr>
<td>3.12</td>
<td>Collective overriding.</td>
<td>123</td>
</tr>
<tr>
<td>3.13</td>
<td>Ranks of the derivations.</td>
<td>124</td>
</tr>
<tr>
<td>3.14</td>
<td>Harmless loop.</td>
<td>126</td>
</tr>
<tr>
<td>3.15</td>
<td>Positive loop.</td>
<td>131</td>
</tr>
<tr>
<td>3.16</td>
<td>Inference rules with finite failure.</td>
<td>133</td>
</tr>
<tr>
<td>3.17</td>
<td>Derivation tree with finite failure.</td>
<td>134</td>
</tr>
<tr>
<td>3.18</td>
<td>Relative size of the derivation relations.</td>
<td>137</td>
</tr>
<tr>
<td>3.19</td>
<td>Bounding the strong derivation relation.</td>
<td>137</td>
</tr>
<tr>
<td>4.1</td>
<td>The fact.</td>
<td>140</td>
</tr>
<tr>
<td>4.2</td>
<td>Positive loop.</td>
<td>142</td>
</tr>
<tr>
<td>4.3</td>
<td>False floating property.</td>
<td>148</td>
</tr>
<tr>
<td>4.4</td>
<td>Simple cycles.</td>
<td>154</td>
</tr>
<tr>
<td>4.5</td>
<td>Multiple rules for the same property in a single class.</td>
<td>156</td>
</tr>
<tr>
<td>4.6</td>
<td>Multiple rules for the same property in different classes.</td>
<td>157</td>
</tr>
<tr>
<td>4.7</td>
<td>Non-minimal arguments.</td>
<td>158</td>
</tr>
<tr>
<td>4.8</td>
<td>Graph of attacks.</td>
<td>159</td>
</tr>
<tr>
<td>4.9</td>
<td>A HST and the corresponding relation attacks.</td>
<td>161</td>
</tr>
<tr>
<td>4.10</td>
<td>Complete extensions.</td>
<td>161</td>
</tr>
<tr>
<td>4.11</td>
<td>Self-overflowing.</td>
<td>162</td>
</tr>
<tr>
<td>4.12</td>
<td>Why the simple embedding fails.</td>
<td>162</td>
</tr>
<tr>
<td>4.13</td>
<td>The extended embedding.</td>
<td>164</td>
</tr>
<tr>
<td>4.14</td>
<td>Another extended embedding.</td>
<td>164</td>
</tr>
<tr>
<td>5.1</td>
<td>The correspondence.</td>
<td>172</td>
</tr>
<tr>
<td>5.2</td>
<td>Translation of double diamond.</td>
<td>173</td>
</tr>
<tr>
<td>5.3</td>
<td>Are these nets ambiguous or not?</td>
<td>177</td>
</tr>
<tr>
<td>5.4</td>
<td>Two HST representations of the <em>boring Clyde</em> example.</td>
<td>178</td>
</tr>
<tr>
<td>5.5</td>
<td>Lions.</td>
<td>179</td>
</tr>
<tr>
<td>6.1</td>
<td>$S_D$ is not continuous.</td>
<td>187</td>
</tr>
</tbody>
</table>
# List of symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma$</td>
<td>inheritance network</td>
<td>4</td>
</tr>
<tr>
<td>$\rightarrow$</td>
<td>positive link in inheritance networks</td>
<td>4</td>
</tr>
<tr>
<td>$\not\rightarrow$</td>
<td>negative link in inheritance networks</td>
<td>4</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>extension of an inheritance network</td>
<td>4</td>
</tr>
<tr>
<td>$\triangleright$</td>
<td>permission relation</td>
<td>6</td>
</tr>
<tr>
<td>$\mathcal{H}$</td>
<td>hierarchically structured theory</td>
<td>14</td>
</tr>
<tr>
<td>$\mathcal{C}$</td>
<td>set of classes</td>
<td>14</td>
</tr>
<tr>
<td>$\mathcal{P}$</td>
<td>set of properties</td>
<td>14</td>
</tr>
<tr>
<td>$\prec$</td>
<td>the hierarchic relation</td>
<td>14</td>
</tr>
<tr>
<td>$\prec, \preceq$</td>
<td>transitive closure of $\prec$</td>
<td>14</td>
</tr>
<tr>
<td>$\mathcal{D}$</td>
<td>class definitions (theories)</td>
<td>14</td>
</tr>
<tr>
<td>$\leftarrow$</td>
<td>explicit negation</td>
<td>14</td>
</tr>
<tr>
<td>$\leftarrow$</td>
<td>conditional</td>
<td>14</td>
</tr>
<tr>
<td>$\overline{p}$</td>
<td>complement of $p$</td>
<td>14</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>filter</td>
<td>14</td>
</tr>
<tr>
<td>$\Rightarrow$</td>
<td>logical implication</td>
<td>14</td>
</tr>
<tr>
<td>$\leftlangle a \rightrangle$</td>
<td>characterization of class $a$</td>
<td>14</td>
</tr>
<tr>
<td>$a:p$</td>
<td>literal (class enjoys property)</td>
<td>15</td>
</tr>
<tr>
<td>$[]$</td>
<td>denotation</td>
<td>17</td>
</tr>
<tr>
<td>$\llangle a \rrangle$</td>
<td>typical subset of $a$</td>
<td>17</td>
</tr>
<tr>
<td>$\mathcal{M}(\mathcal{H})$</td>
<td>set of models of $\mathcal{H}$</td>
<td>19</td>
</tr>
<tr>
<td>$M_\mathcal{H}$</td>
<td>extensional standard model of $\mathcal{H}$</td>
<td>24</td>
</tr>
<tr>
<td>$\models$</td>
<td>satisfaction; entailment</td>
<td>26</td>
</tr>
<tr>
<td>$t$</td>
<td>true</td>
<td>39</td>
</tr>
<tr>
<td>$u$</td>
<td>undefined</td>
<td>39</td>
</tr>
<tr>
<td>$f$</td>
<td>false</td>
<td>39</td>
</tr>
<tr>
<td>$I$</td>
<td>interpretation</td>
<td>47</td>
</tr>
<tr>
<td>$\Delta$</td>
<td>set of defaults</td>
<td>48</td>
</tr>
<tr>
<td>$T_p(\Delta)(I)$</td>
<td>immediate consequence in predicate inheritance</td>
<td>48</td>
</tr>
<tr>
<td>$T\upharpoonright n$</td>
<td>iterator</td>
<td>49</td>
</tr>
<tr>
<td>Symbol</td>
<td>Definition</td>
<td>Page</td>
</tr>
<tr>
<td>---------------</td>
<td>-----------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>$T^c_p(\Delta)$</td>
<td>consequences in predicate inheritance</td>
<td>49</td>
</tr>
<tr>
<td>$S^a(\Delta)$</td>
<td>Scenario</td>
<td>49</td>
</tr>
<tr>
<td>$(\Delta, \Gamma_c(\Delta))$</td>
<td>Split scenario</td>
<td>54</td>
</tr>
<tr>
<td>$\Gamma_c(\Delta)$</td>
<td>consequences in a scenario</td>
<td>54</td>
</tr>
<tr>
<td>$D^c_\cdot$</td>
<td>composed theory</td>
<td>65</td>
</tr>
<tr>
<td>$D$</td>
<td>family of relevant class definitions</td>
<td>73</td>
</tr>
<tr>
<td>$T_D(I)$</td>
<td>immediate consequence in rule inheritance</td>
<td>73</td>
</tr>
<tr>
<td>$T^r(D)$</td>
<td>consequences in rule inheritance</td>
<td>73</td>
</tr>
<tr>
<td>$D[I]$</td>
<td>restriction</td>
<td>73</td>
</tr>
<tr>
<td>$\Delta_D$</td>
<td>increment</td>
<td>73</td>
</tr>
<tr>
<td>$D/J$</td>
<td>quotient</td>
<td>73</td>
</tr>
<tr>
<td>$S_D(J)$</td>
<td>safe conclusions</td>
<td>73</td>
</tr>
<tr>
<td>$C(A)$</td>
<td>inference operation</td>
<td>83</td>
</tr>
<tr>
<td>$\bar{D}$</td>
<td>residual theories</td>
<td>83</td>
</tr>
<tr>
<td>$&lt;\overline{a}&gt;$</td>
<td>weak characterization</td>
<td>83</td>
</tr>
<tr>
<td>$&lt;a&gt;_o$</td>
<td>credulous characterization</td>
<td>87</td>
</tr>
<tr>
<td>$\vdash_a$</td>
<td>strong derivation relation</td>
<td>96</td>
</tr>
<tr>
<td>$\vdash_w$</td>
<td>weak derivation relation</td>
<td>96</td>
</tr>
<tr>
<td>$\vdash_b$</td>
<td>local derivation relation</td>
<td>96</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>derivation relation</td>
<td>97</td>
</tr>
<tr>
<td>$\mathcal{H}^*$</td>
<td>logic program translated from a $\mathcal{H}$</td>
<td>103</td>
</tr>
<tr>
<td>$\mathcal{M}$</td>
<td>well-founded model, class model</td>
<td>109, 19</td>
</tr>
<tr>
<td>$\Gamma_p(I)$</td>
<td>stability operator of a logic program</td>
<td>113</td>
</tr>
<tr>
<td>$\subseteq$</td>
<td>order relation on triples and on class models</td>
<td>119, 140</td>
</tr>
<tr>
<td>$\leftarrow$</td>
<td>dependency</td>
<td>119</td>
</tr>
<tr>
<td>rank()</td>
<td>rank of a derivation</td>
<td>122</td>
</tr>
<tr>
<td>$\models_a$</td>
<td>strong derivation relation with finite failure</td>
<td>133</td>
</tr>
<tr>
<td>$\models_w$</td>
<td>weak finite failure relation</td>
<td>133</td>
</tr>
<tr>
<td>$\models_b$</td>
<td>local derivation relation with finite failure</td>
<td>133</td>
</tr>
<tr>
<td>$I$</td>
<td>class interpretation</td>
<td>139</td>
</tr>
<tr>
<td>$S$</td>
<td>strong component of an interpretation</td>
<td>139</td>
</tr>
<tr>
<td>$W$</td>
<td>weak component of an interpretation</td>
<td>139</td>
</tr>
<tr>
<td>$T_B^g$</td>
<td>weak immediate consequence</td>
<td>142</td>
</tr>
<tr>
<td>$T_D^g$</td>
<td>strong immediate consequence</td>
<td>142</td>
</tr>
<tr>
<td>$\Sigma_D$</td>
<td>stability operator</td>
<td>142</td>
</tr>
<tr>
<td>$\cap$</td>
<td>intersection of interpretations</td>
<td>147</td>
</tr>
<tr>
<td>$AF$</td>
<td>argumentation framework</td>
<td>151</td>
</tr>
<tr>
<td>$AR$</td>
<td>set of arguments</td>
<td>151</td>
</tr>
<tr>
<td>attacks</td>
<td>relation of attacks</td>
<td>151</td>
</tr>
<tr>
<td>$F_{AF}(S)$</td>
<td>characteristic function</td>
<td>152</td>
</tr>
<tr>
<td>$GE$</td>
<td>grounded extension</td>
<td>152</td>
</tr>
<tr>
<td>$(S, b-q)$</td>
<td>argument with support $S$ and head $b-q$</td>
<td>155</td>
</tr>
</tbody>
</table>
Chapter 0

Introduction

Axiom I

If two contrary actions be excited in the same subject, a change must take place in both, or in one alone, until they cease to be contrary.

in Benedict de Spinoza, Ethic, Fifth part: of the power of the intellect, or the human liberty.

0.1 Subject delimitation

The starting point for this work is the recognition of the importance of taxonomies in the systematization of world knowledge. A common practice in scientific activity is the search for classifications established by recognizing common properties among individuals of a chosen domain. This induces the definition of taxonomies structured as hierarchies of classes with associated properties, from which information can be retrieved with the help of a more or less formalized kind of inheritance reasoning. Taxonomies have a definitional flavor but are not always accurate, implying a provision for special cases or exceptions. Usually, disagreements can be modeled in terms of specific properties, without affecting the taxonomic core. A whale is a mammal by definition but is odd in that most mammals are land-dwellers.

Besides predicate logic, also semantic networks and frame systems have been widely used to represent knowledge that can be factored at different levels of specificity and shared, thus avoiding redundancy. As a particular case of semantic nets, inheritance networks have attracted a special attention.

The main trend in inheritance reasoning is the inheritance network proposal, where the basic elements are nodes and links. Nodes represent individuals and defeasible properties. Links are of two kinds: positive, interpreted as is-a [Bra82], and negative, meaning is-not-a. In these systems, there is no ontological distinction between classes and properties. However, in natural languages there is always a
fundamental distinction between nouns (classes) and adjectives/verbs (properties), which seems to be an important characteristic that a knowledge representation system should possess in order to be natural.

Inheritance of properties is related to the existence of paths along positive links, possibly except the last, with certain characteristics determined by the way specificity is understood. Negative links express both the inheritance of a negative property and the blocking of further reasoning along that path. So, negative links have a different status and are mainly used to represent exceptions. This induces a bias in the choice of which properties are to be presented in a positive way and which will be exceptions. This lack of symmetry may be unpleasant from a knowledge representation viewpoint. Paths in inheritance networks play a double role: supporting the association of properties to individuals and inducing a specificity relation. The two roles interfere sometimes in undesirable ways. The present work studies reasoning in a system where they are clearly separated.

Class/property systems seem an appropriate way to represent natural kinds. Such systems were formalized as a particular case of inheritance networks in [Tou86]. But the subcase possessing interesting computational properties, the orthogonal class/property systems, is too restricted and limited to simple forms of nonmonotonicity. It is not able to express dependencies between properties, one of the motivations of the present work.

As rules can represent dependencies, structured Logic Programming [MP89, MP92] is an appealing framework to develop an inheritance system, leading to a straightforward parallel between modules/classes and predicates/properties.

In our proposal, specificity is firmly established by a hierarchic relation over classes. Each class possesses a theory stating its local properties either directly or implicitly through rules. Properties holding for the class are obtained from its theory composed with those inherited from the superclasses. Properties may be positive or negative. So contradictions may arise, specially because the language allows multiple inheritance. If one of the contradictory properties is stated in a more specific class, it overrides the other one, this way modeling an exception. If none is more specific than the other, then there is a situation of ambiguity whose solution is the mutual neutralization of the opponents. So, properties constitute the defeasible part of the system.

The interest in modeling exceptions and ambiguities led to the adoption of a partial or 3-valued logic framework along the lines of the work of [GL90], which is directly addressed to extend Logic Programming with explicit negation. It also departs from the traditional Closed World Assumption [Rei80] that is present even in other 3-valued semantics, like the Well Founded [VGRS90], as a preference for negative assumptions. We adopt the position of minimizing information, not truth, in the sense that our positive or negative conclusions are explicitly supported by
the program. Everything else is considered undefined.

In [GL90] the appearance of a contradiction spreads to the whole model: every literal becomes true, which amounts to annihilate its information content. This seems to be meaningful in the context of a single theory. In our proposal, contradictions caused by multiple inheritance are just defaults and so may be fought locally by setting the intervening literals to undefined.

In a flat, i.e., not hierarchic, Logic Programming system, negation as failure is often used to model a default policy [KS90]. But then positive and negative literals are not symmetric. Negative literals represent exceptions and so they are preferred, with the purpose of obtaining a reduction of programs to the standard SLDNF case. Our goal is to rely on the structure represented by the hierarchy to specify as defaults properties inherited from superclasses. For that reason we deliberately omitted the negation as failure operator.

Also motivated by the same spirit of concentrating on the specific problems of inheritance, the language is restricted to the propositional case, or Datalog with negation.

The scope of this work is the characterization of the set of consequences which should be expected of a knowledge base structured as a taxonomy of classes with associated theories. This set is intended to contain just the indisputable or skeptical conclusions. At the core of our solution is the use of another set of properties serving as a counterpoint. It contains all the properties which have a reason to be true, given the skeptical set. It has thus a credulous essence. The phenomenon of convergent oscillation between the two sets is present in the several perspectives under which we will analyze the hierarchic system. At some points we will use, for clarity, a third set that is an emanation of the skeptical set and records its incidence in each class.

The overall goal is to contribute to clarify aspects of the taxonomic fragment of knowledge based systems, in particular the notions of inheritance and dependency among properties and classes, which are mixed together in the inheritance networks formalism, while also providing more expressiveness. The communication between the taxonomic subsystem and the other components of a knowledge base is done through goals of the form \( c::p \), which means that class \( c \) enjoys property \( p \).

Inheritance is also a main ingredient of object-oriented systems. It is possible to establish a parallel between our classes and objects. But, instead of methods and actions, we have conditional properties. We may contrast two views of object-orientation:

**objects as named theories** An object is a set of rules, or a set of clauses grouped around an implicit first argument, instead of around the predicate name - this is the case of our proposal.
complex objects An object is a composition, using appropriate operators, of simpler objects - this better describes concept languages.

The former lives well with the notion of defeasible properties, while keeping the strict side as a simple is-a relation. The latter elaborates on the strict side, sees the is-a as the result of inference, but has more difficulties with the defeasible side. Combining both requires a clear statement of the meaning of specificity. It must be dynamically inferred before being used by the defeasible part. We suspect that it may require an infinite number of classes, to represent all the possible concepts.

We do not explore further this line of research. Neither will we pursue the relationship with the probabilistic approach [Neu90, Neu91], based on the relative plausibility of premises and conclusions. We will concentrate on the problems addressed by inheritance reasoning and make use of the techniques developed in logic programming.

0.2 Inheritance networks

Inheritance networks are one of the most influential approaches to inheritance reasoning due to the naturalness and simplicity of its representation. However, inference from the nets is not so intuitive as evidenced by the diversity of proposals [THT87]. In [San86] it was even proposed that the semantics of nets could be stated via a catalog of canonical examples, classified by humans and serving as a pattern against which implementations should be tested. In this short introduction to path-based inference in inheritance networks we follow the formulation of [TTH91].

An inheritance network \( \Gamma \) is a set of positive and negative links of the form \( x \rightarrow y \) and \( x \not\rightarrow y \), respectively. Links in \( \Gamma \) determine a graph, which must be acyclic. If a certain node \( x \) is the end node of no link then it is an individual, otherwise it is a property.

A path \( \sigma \) is a sequence of links. It is positive or negative according to the kind of the last link. All the other links must be positive. We will use lower case Greek letters for paths and Latin letters for nodes. To ease the notation, we allow subpaths like \( \tau \) in \( x \rightarrow \tau \rightarrow y \) to be empty, a node or a sequence of links.

An extension \( \Phi \) of a net \( \Gamma \) is a set of paths containing \( \Gamma \) and all the paths permitted by some inheritance method. With respect to an extension \( \Phi \), a net \( \Gamma \) permits a path \( \sigma \), \( \Gamma \Permits \sigma \), if and only if \( \sigma \in \Phi \). A common requirement for the inheritance method is that it must not permit, in a single extension, contradictory paths, \( x \rightarrow \tau \rightarrow y \) and \( x \rightarrow \sigma \not\rightarrow y \), unless the net itself is contradictory, i.e., it contains two links \( w \rightarrow z \) and \( w \not\rightarrow z \). Another criterion is that most specific information takes precedence.
0.2 Inheritance networks

An algorithm to compute the extension could be based on a shortest path wins principle. In the first step it permits every link. So, no compound path can conflict a link. The paths permitted by this algorithm are not always intuitive, as can be seen in example 1.

Example 1 Shortest-path method.

\[
\begin{aligned}
    &\text{Figure 0.1: Clyde, the elephant.} \\
    &\text{In the interpretation } c = \text{Clyde}, r = \text{royal-elephant}, e = \text{elephant}, g = \text{gray}, \\
    &\text{royal-elephants are a special kind of elephants which are not gray. An extension } \\
    &\Phi \text{ for the net does not contain the compound path } r \rightarrow e \rightarrow g \text{ because it conflicts} \\
    &\text{with the negative link } r \not\rightarrow g \text{ and links belong to } \Phi \text{ by definition.} \\
    &\text{The paths } \sigma = c \rightarrow r \not\rightarrow g \text{ and } \tau = c \rightarrow r \rightarrow e \rightarrow g \text{ conflict. Using a shortest } \\
    &\text{path algorithm, } \sigma \text{ wins from } \tau. \text{ However the net contains the redundant statement} \\
    &\text{that Clyde is an elephant and thus the path } c \rightarrow e \rightarrow g \text{ conflicts } \sigma \text{ and has the} \\
    &\text{same length, configuring a tie. This is at odds with the preference for more specific} \\
    &\text{information, as royal-elephants are elephants.} \\
    &\square
\end{aligned}
\]

The solution in [Tou86] is to order paths not according to a measure of length but to a measure of “between-ness”, which is called inferential distance. In the example, the existence of the path \(c \rightarrow r \rightarrow e\) makes \(r\) inferentially closer to \(c\) than \(e\) and so gives precedence to \(c \rightarrow r \not\rightarrow g\) over \(c \rightarrow e \rightarrow g\).

Inferential distance is at the heart of the following definitions.

Definition 1 (Sitiator) A path \(\rho\) is a sitiator of \(x \rightarrow \tau \rightarrow w \not\rightarrow y\) with respect to \(x \rightarrow \sigma \rightarrow z \rightarrow y\) if and only if \(\rho\) has the form \(x \rightarrow \tau_1 \rightarrow w \rightarrow \tau_2 \rightarrow z\). (Similarly for paths of opposite polarity.)

If the sitiator is permitted then node \(w\) is between \(x\) and \(z\). This definition corresponds to off-path preemption [San86].

Definition 2 (Preemption) A path \(x \rightarrow \sigma \rightarrow z \rightarrow y\) is preempted in \(\Gamma\) if and only if \(\Gamma \vdash x \rightarrow \sigma \rightarrow z\) and there is a preemtpt path \(x \rightarrow \tau \rightarrow w \not\rightarrow y\) with a permitted sitiator. (Similarly for paths of opposite polarity.)
Preemption is a form of overriding. A confictor of a path \( x \rightarrow \sigma \rightarrow y \) is a path \( x \rightarrow \tau \not\equiv y \) having no permitted situator with respect to the former path. A contradictory link \( x \not\equiv y \) is seen as a preemperor because it is always situated with respect to any path starting at \( x \).

**Definition 3 (Conflict)** A path \( x \rightarrow \sigma \rightarrow y \) is conflicted in \( \Gamma \) if and only if \( \Gamma \models x \rightarrow \sigma \) and there is a confictor \( x \rightarrow \tau \not\equiv y \) such that \( \Gamma \models x \rightarrow \tau \). (Similarly for paths of opposite polarity.)

Conflicts in a network may either generate alternative extensions, in a credulous standing, or no conclusion, in the skeptical extension. Confictors and situated preemptors are defeaters.

**Definition 4 (Inheritability)** A path \( x \rightarrow \tau \rightarrow z \rightarrow y \) is inheritable if and only if \( \Gamma \models x \rightarrow \tau \rightarrow z \) and \( z \rightarrow y \in \Gamma \). (Similarly for paths of opposite polarity.)

Paths are built upwards, adding links to the end of permitted paths. Other reasoners are downwards or even bidirectional, requiring \( \Gamma \models x \rightarrow \tau \) and \( \Gamma \models \tau \rightarrow z \) to get \( \Gamma \models x \rightarrow \tau \rightarrow z \).

**Definition 5 (Permission)** A path \( \sigma \) is permitted, \( \Gamma \models \sigma \), if and only if \( \sigma \in \Gamma \) or \( \sigma \) is inheritable and has no defeaters.

There is an algorithm [HTT90] to compute the upwards skeptical extension of a consistent net. A generalized path is a sequence of indistinctly positive or negative links. The degree of a path \( \sigma \) is the maximum number of links of any generalized path between the initial and the end nodes of \( \sigma \). The algorithm initializes the extension with the net and inductively adds to it the permitted paths of successively higher degree. Notice that the permission of a path of degree \( n + 1 \) depends only on the permission of paths of degree \( n \).

**Definition 6 (Credulous permission)** A path \( \sigma \) is permitted if \( \sigma \in \Gamma \) or \( \sigma \) is inheritable and has no preemptors.

Absence of conflictors is not required for credulous reasoning. It is enough to guarantee consistency of the extension. An upwards credulous extension [Tou86] is a maximal consistent set of (credulous) permitted paths. In the sequel, if not stated otherwise, skeptical permission is assumed.

Lifting the requirement of absence of conflictors opens the door to alternative extensions whenever a conflict exists.
Example 2 *Skeptical and credulous extensions.*

The net in figure 0.2 is borrowed from [HTT90]. The skeptical extension $\Phi$ is obtained after two iterations.

\[
\begin{align*}
\Phi^1 &= \Gamma \\
\Phi^2 &= \Phi^1 \cup \{n \rightarrow r \rightarrow f, r \rightarrow f \not\rightarrow a, q \rightarrow p \rightarrow a\} \\
\Phi^3 &= \Phi^2 \cup \{n \rightarrow r \rightarrow f \not\rightarrow a\} = \Phi
\end{align*}
\]

The credulous extensions are

\[
\begin{align*}
\Phi_a &= \Gamma \cup \{n \rightarrow r \rightarrow f, r \rightarrow f \not\rightarrow a, q \rightarrow p \rightarrow a, \\
&\quad n \rightarrow q \rightarrow p, n \rightarrow q \rightarrow p \rightarrow a\} \\
\Phi_b &= \Gamma \cup \{n \rightarrow r \rightarrow f, r \rightarrow f \not\rightarrow a, q \rightarrow p \rightarrow a, \\
&\quad n \rightarrow q \rightarrow p, n \rightarrow r \rightarrow f \not\rightarrow a\} \\
\Phi_c &= \Gamma \cup \{n \rightarrow r \rightarrow f, r \rightarrow f \not\rightarrow a, q \rightarrow p \rightarrow a, \\
&\quad n \rightarrow r \not\rightarrow p, n \rightarrow r \rightarrow f \not\rightarrow a\}
\end{align*}
\]

It is curious that the path $\rho = n \rightarrow r \rightarrow f \not\rightarrow a$ is in the skeptical extension while it is not present in every credulous extension. The problem lies in that the confictor $n \rightarrow q \rightarrow p \rightarrow a$ does not actually conflict with $\rho$ because its initial segment $n \rightarrow q \rightarrow p$ is conflicted by $n \rightarrow r \not\rightarrow p$. This first ambiguity is not propagated and path $\rho$ opportunistically becomes permitted. We will return to this subject in chapter 5.

Definition 7 (Conclusions) *The conclusions of a net with respect to an extension $\Phi$ are the pairs $x \rightarrow y$ (resp. $x \not\rightarrow y$) such that $x \rightarrow \tau \rightarrow y$ (resp. $x \rightarrow \tau \not\rightarrow y$) is a path in $\Phi*.\]
0.3 Original contributions of the thesis

The problem solved by this thesis is the precise statement of inheritance in a hierarchy of extended logic programs. It tries to overcome the deficiencies of inheritance networks, some of which were hinted at in the previous section, both at the language level and from the semantic viewpoint.

The language is significantly more expressive than that of inheritance networks, in particular in the following points:

- It is able to express definitional statements, besides defeasible assertions.
- It allows dependencies on conjunctive conditions.
- Reasoning can proceed from a negative conclusion. Therefore, the representation of knowledge does not get biased by unnatural asymmetries.

The semantics is more declarative, directly based on the logic notion of contradiction. There is an extensional semantics in which alternative inheritance mechanisms may be plugged in and a fixed-point semantics characterizing two of them. Predicate inheritance is shown to have severe shortcomings. Rule inheritance is defined by a constructive bottom-up operator and also studied from the viewpoint of a more declarative stable semantics.

The obtained solution belongs to the skeptical family and is of the ambiguity propagation kind. It does not surpass the ideal skeptical semantics defined by the intersection of all the credulous extensions, as opposed to the main trend of inheritance networks whose ambiguity blocking inference algorithm does not guarantee that. To achieve this result two complementary relations are required. The strong relation, which embodies the skeptical semantics, and the weak relation, which combines the several credulous alternatives. The two relations are mutually defined through negation.

The existence, uniqueness, and consistency of the semantics are formally proved.

The semantic results acquire a special interest if an implementation is feasible. The thesis presents an operational semantics which is goal-directed and provides some hints for an effective implementation. This is illustrated by a naive example of a Prolog interpreter for the language.

A modular translation to Logic Programming is defined which implements the directly skeptical semantics. To our knowledge, no previous attempt to formalize the skeptical version of inheritance networks, using either Default Logic or Logic Programming, has been able to do that in a modular way.

The thesis includes an embedding of the developed solution in the Argumentation Theory. It constitutes another example of the generality of that theory as
a tool to study a wide range of nonmonotonic systems and of the ability of Logic Programming as an inference machine to implement them.

0.4 Outline of the thesis

Chapter 1 is devoted to the presentation of the proposed language and of its extensional semantics. The language is kept as simple as possible to favor concentration on the problems of inheritance. However, it is shown that it can be easily enriched without affecting the core of the work. There is also a discussion of three-valued and complete information approaches. The extensional semantics is built from a domain of individuals. Classes and properties meet at this level, as they both denote sets of individuals. In the remaining chapters the diverse roles they play become more apparent, with the dynamics of properties catching most of the attention. The extensional semantics assumes that the set of properties implicit in the theories of a class and its superclasses is known. The method to make these sets explicit is the subject of the following chapters.

Chapter 2 presents two different methods. In the first, of a more local nature, inheritance applies recursively to the conclusions of the immediate superclasses. In the second, the rules themselves are inherited allowing a more sensible behavior, at the cost of loosing compositionality when computing inheritance with respect to a class. After a detailed motivation, an iterated fixed-point operator is defined, which embodies a skeptical semantics. The inner iteration is based on a restricted version of the immediate consequence operator that is aware of overriding. The outer iteration copes with neutralization. The chapter explores some characteristics of rule inheritance, paying special attention to how it deals with ambiguity. In this context, a credulous semantics is also proposed, which is subsumed by the just mentioned inner iteration.

In contrast with the argument construction approach of chapter 2, chapter 3 is goal-oriented. The inference system reflects the oscillation between two successively more close extremes, present in the bottom-up operator, by simultaneously defining two, weak and strong, inference relations. The strong relation actually has two forms, as a global conclusion of the system and as a local inference in the class where the corresponding rule is stated. The mutual negative dependence between the strong and weak relations makes the soundness of one rely on the completeness of the other. To give an insight on the implications of the chosen formalism, the inference system is translated to a logic program. As a by-product, the translation induced a naive implementation of a meta-interpreter for the language, listed in annex A. The well-founded semantics of the translated program agrees with the strong inference relation. The chapter proceeds with the proofs of consistency of the operational semantics and of its soundness and completeness with respect to
the fixed-point semantics. It concludes with an alternative inference system that explicitly defines the finite failure inference rule for negation.

Chapter 4 has two parts. In the first it is argued that the notion of model is too weak to be useful and even supported models are of little significance if the theories contain positive loops. Stable models overcome the difficulties and the least stable model coincides with the least fixed-point of the bottom-up operator. In the second part, and after a brief introduction to the argumentation theory, it is shown that this unifying framework provides an appropriate level of abstraction to study the nonmonotonic behavior of the inheritance mechanism.

Chapter 5 is devoted to a comparison with related work on inheritance reasoning, in particular with the path-based analysis of inheritance networks. Its skeptical ambiguity-blocking semantics can be approximated by a variation of the inference system. The chapter concludes with some comments on the expressivity of the discussed proposals.

The thesis ends with the small chapter 6, where its conclusions are summarized and some remarks on possible developments are gathered.
Chapter 1

The framework: syntax and extensional semantics

The chapter begins with a motivation of the intuitions underlying the language proposed. Follows a basic version of the language and an extensional semantics. To conclude, there is a discussion of several extensions and alternatives both to the language and to the semantics.

1.1 Hierarchically structured theories

1.1.1 Presentation

In our proposal, a system is a taxonomic structure of classes. Class inclusion is strict and has a definitional flavor [Bra85]. This is the non-defeasible part of the system. Each class has a local associated theory from which its specific properties are obtained. The superclasses' theories provide properties which are inherited by the class, when they do not clash with more specific information. Properties constitute the defeasible part of the system. They behave as defaults which may be overridden by a contradictory property asserted in a subclass, or neutralized (along) with a contradictory property in another superclass.

Rule heads in the theories may be positive or negative literals, and these have the same status. Negation is not used to block further reasoning, as in inheritance nets, but only to assert the non-property. Exceptions are obtained directly from the notion of contradiction\(^1\), favoring the conclusion stated in the more specific class, no matter whether positive or negative. Heads are not disjunctive to allow for a constructive inference directed to the head literals. So rules are seen as (partially) defining properties (those in their heads).

\(^1\)In Logic Programming, nonmonotonicity is based on failure to prove, instead.
A negative conclusion explicitly asserts the corresponding atomic property to be false. Failure to prove a literal and its opposite assigns the value undefined to the corresponding atom, and the same does a contradiction (proving both) originated in two classes that are non-comparable, i.e., neither of which is more specific than the other. In order to get a symmetric treatment of both positive and negative information, it is also needed that reasoning may proceed from a negative conclusion. This arises naturally by allowing negative literals in rule bodies.

**Example 3** The bat hierarchy.

Consider the following assertions:

1. Bats are mammals.
2. Mammals are animals.
3. Non-flying mammals are quadruped.
4. Flying animals are light.
5. Mammals do not fly.

Taking the first two assertions as definitions and the other four as normative but allowing for exceptions, a possible representation, in our formalism, is displayed on the left of Fig. 1.1. Rounded-corner boxes represent classes which are linked by *isa* arrows. The rectangles contain the class definitions, which are theories consisting of property definitions. The two rules in *mammal* support the conclusion *quadruped* there, whereas in *bat* what is inherited is the rule for *quadruped* which has no consequences since its premise is false.
The inheritance network on the right represents the same information, but suffers from two problems. To represent assertion 3 we need a positive property for non-flying ($nf$), allowing a path to quadruped. This node has no relationship with the node $f$, inside the system, so an external integrity constraint must be added insuring that both properties are interpreted as the negation of each other, i.e., they are not simultaneously true for any individual. To enforce such a constraint redundant links are needed, creating a potential problem of inconsistency. Two links are used to represent assertion 5, saying that mammals aren't flying and are non-flying.

We argue that, from a knowledge representation viewpoint, the separation into a strict and a defeasible part provides greater expressiveness because it allows perennial (definitional) truths to be stated in a non-revisable form, as a hierarchy of classes. We also have the power to represent natural kinds, which admit exceptions, through defeasible properties. There is however a certain duality between a class and its defining property, which is patent in the semantics of the system. Both classes and properties denote sets of individuals. But while every individual in a class must belong to all its superclasses, it is not forced to enjoy all the properties associated with them, i.e., to belong to the intersection of all such properties. What is then the meaning of saying that a set of properties characterizes a class? The answer involves the definition of a second denotation for classes, again a set of individuals, included in the previous one, which contains the typical individuals for the class, those individuals which do verify all the classes’ properties. So, the order relation in the taxonomy imposes the inclusion of the classes, but the associated theories support conclusions which have to be true only for the corresponding typical subsets. Atypical individuals which are exceptions with respect to certain properties enjoyed by the class may still belong to it. A bat is a mammal indeed, but it is an exceptional (not typical) flying mammal.

The problem of finding which properties hold for each class is previous to the interpretation in terms of sets of individuals. This account of the semantics is a framework where several inheritance mechanisms may be plugged in, shaping the system in different ways. We will briefly discuss predicate inheritance and present in a detailed way the more powerful rule inheritance.

Another advantage of our system is its ability to deal with dependencies on multiple premises, through conjunction in the bodies. In the example, assertion 3 becomes more accurate if rephrased as non-flying and non-swimming mammals are quadruped, with the obvious representation quadruped ← ~flying, ~swimming. Inheritance networks are not able to directly represent conjunctive links.
The framework: syntax and extensional semantics

1.1.2 The language of HST

A Hierarchically Structured Theory (HST) is a partial order of classes with associated theories. Specificity is governed by the order relation and is strict. The theories define properties. These may be positive or negative and are defeasible. Theories assert facts (simple properties) and rules stating dependencies among properties.

Definition 8 (HST) A Hierarchically Structured Theory $\mathcal{H} = (C, \mathcal{P}^+, <, \mathcal{D})$ is a 4-tuple where

- $C$ is the set of classes;
- $\mathcal{P}^+$ is the set of positive properties;
- $<$ is the hierarchic relation, which must be an acyclic relation on $C$;
- $\mathcal{D} : C \rightarrow \varphi(\mathcal{R})$ is a total function called class definition mapping each class to a set of property rules.

The hierarchic relation $<$ is intended to state just the direct sub/superclass relation. We will also use $<$ and $\leq$ for its transitive and, resp. reflexive transitive closures. As no restriction is put on $<$ other than being acyclic, a class may have several direct superclasses, thus supporting multiple inheritance.

The set of properties is $\mathcal{P} = \mathcal{P}^+ \cup \mathcal{P}^-$ where $\mathcal{P}^+$ contains the atomic properties and the set $\mathcal{P}^- = \{\neg p | p \in \mathcal{P}^+\}$ contains their negations. Properties are specified through rules. The set of all rules is

$$\mathcal{R} = \{p \leftarrow Q | p \in \mathcal{P}, Q \subseteq \mathcal{P}\}.$$ 

Each rule $p \leftarrow Q$ specifies a condition $Q$ for $p$ to hold, so it is said to be a rule for $p$. The bodies of property rules are sets. In the following, we will use the notation $x, X = \{x\} \cup X$, s.t. $x \notin X$ for the nondeterministic choice of an element in a set. The function $\mathcal{D} : C \rightarrow \varphi(\mathcal{R})$ maps each class $a$ to a set of rules $\mathcal{D}_a$ said to be locally defined in $a$. $\mathcal{D}_a$ will also be sometimes called the theory associated to class $a$. A rule $p \leftarrow \emptyset$ amounts to asserting $p$ and will be called a fact.

Complementation is defined in the usual way:

$$p \in \mathcal{P}^+ \Rightarrow \overline{p} = \neg p$$
$$\neg p \in \mathcal{P}^- \Rightarrow \overline{\overline{p}} = p$$

$$Q \subseteq \mathcal{P} \Rightarrow \overline{Q} = \{\overline{p} | p \in Q\}.$$ (1.1)
Definition 9 (Contradiction) A set of properties $X$ is said to be contradictory (or inconsistent) if and only if it contains a complementary pair, $\{p, \lnot p\} \subseteq X$.

Definition 10 (Filter) We call filter to the unary operator $\Theta : \wp(\mathcal{P}) \rightarrow \wp(\mathcal{P})$, which gives the non-contradictory subset of a set of properties,

$$\Theta X = X - \overline{X}.$$ 

Apart from the hierarchic relation, the sole kind of information expressible in the language is what properties are attributable to each class. These properties characterize the expectable common behavior of the individuals in the class, discounting the exceptions. We will thus call the characterization of a class $a$ (written $<a>$) to the set of properties that hold for that class. The properties are implicit in the theories associated to the class and to its superclasses. So, they may be either locally specified or inherited via an appropriate inheritance mechanism.

Making the actual contents of a characterization explicit is the main topic of this thesis. It depends on the specific kind of inheritance considered and will be defined in the following chapters. In the current chapter, we will establish a semantics for the HST that is parametric on the characterization. It acts as a framework in which different versions of the inheritance mechanism may be plugged in.

In the present study we are concentrating on the potential of the system to factor out knowledge and express default properties guided by the hierarchy. However, this can be considered as a subsystem of a larger knowledge base, possibly offering other knowledge representation methods, which communicates with the taxonomic part through statements of the form $a : p$, which are true if class $a$ enjoys property $p$. Our purpose is to establish how to find out if the property $p$ belongs to the characterization $<a>$ of class $a$. So the property does not have a logic value by itself but only relatively to a class. Abusing language we will however talk about property rules as if they were clauses, by considering each property as a literal abstracted on its first argument, the target class, when there is no danger of confusion.

The language above is very simple. Although it is powerful enough for the purposes of this work, it could be easily enriched in two directions.

The first is to allow properties to become first-order entities. An atomic property $p \in \mathcal{P}^+$ is of the form $n(t_1, \ldots, t_m)$, where $n$ is a property name (equipped with the arity $m$) and each $t_i$ is a term. Terms are defined in the usual way out of a vocabulary of constants, variables, and function symbols used in the rules of the HST. The vocabulary is unique for the whole hierarchy to give a common language to all the classes and is disjoint from $\mathcal{C}$ and $\mathcal{P}^2$. The set of all ground instances of

---

2It is an interesting subject, deserving further investigation, to consider the case where $\mathcal{C}$ is contained in the constants of the vocabulary. This allows the expression of inheritable relations
terms is the Herbrand universe. Non-instantiated terms may be allowed in property arguments. Rules containing such terms are considered universally quantified and taken as a short hand for their full instantiation over the Herbrand universe. If no function symbols are allowed and the set of constants is finite (Datalog case) then the instantiated program is finite and so it is still essentially propositional. Otherwise, infinite programs arise.

The other extension to the language is the introduction of more connectives. Besides conjunction, formulas might include constructive disjunction and negation, along the lines of the logic of strong negation [Wag90]. We will return to this subject in the next section.

1.2 Extensional semantics

1.2.1 Interpretations

A pre-interpretation \( I \) of a HST consists of a domain of individuals \( \mathcal{U}_I \) and three interpretation functions for classes and for properties:

\[
\begin{align*}
\kappa_I &: \mathcal{C} \rightarrow \wp(\mathcal{U}_I) \\
\pi_I &: \mathcal{P} \rightarrow \wp(\mathcal{U}_I) \\
\tau_I &: \mathcal{C} \rightarrow \wp(\mathcal{U}_I)
\end{align*}
\]

The extension of a class is the set of individuals into which the class is mapped by the function \( \kappa_I \). A property is mapped, by the function \( \pi_I \), to a set of individuals, too. This coincidence reflects the duality between extension (class) and intension (its defining properties) of a set. A certain inter-mutability, at the knowledge representation level, between classes and properties enforces the choice of the same image space for both interpretation functions.

In the sequel, the typical subset of a class will be related to the class itself. However, we want to bring to the semantic level the intuitive notion of individuals which are typical of a certain class, in the sense that all the characteristic properties of the class hold for those individuals. This is the reason why we provide a second interpretation function (\( \tau_I \)) for classes.

Individuals exist only at the semantic level. In the language there are just classes and properties, for the sake of uniformity. This is not a limitation from the knowledge representation viewpoint because one may define classes to any level of detail, until there is a single "individual" denoted by the class. As long as there among classes, a far more intricate matter than the inheritance of simple properties associated to classes.
are properties allowing for a distinction, for example the individual's name, the refinement of the hierarchy is meaningful.

The language does not support properties applied globally to a class, rather than to each of its individuals, like the cardinality of the class. It is not able to make assertions about sets of individuals.

The denotations of classes and properties are defined directly from the pre-interpretation functions:

\[(\text{taxonomic})\quad c \in C \Rightarrow [c]_I = \kappa_I(c)\]
\[(\text{property})\quad p \in P \Rightarrow [p]_I = \pi_I(p)\]
\[(\text{typical})\quad c \in C \Rightarrow \llangle c \rrangle_I = \tau_I(c).\]  

Sets of properties are interpreted conjunctively. The denotation of a set is the intersection of the denotations of all the properties in it:

\[\varnothing)_I = \mathcal{U}_I\]
\[\left\{ [p] \cup [Q] \right\}_I = [p]_I \cap [Q]_I.\]  

We have already introduced an important set of properties which is the characterization of a class. It contains all the properties which typically hold for the class and is built, by an appropriate inheritance mechanism, out of the theories associated to the class and its superclasses.

**Example 4** A simple inheritance mechanism.

If the HST is restricted to the case where theories cannot contain rules, i.e., they are sets of properties, the following is a possible definition of the characterization of a class a:

\[\llangle a \rrangle = \Theta \left( \bigcup_{a \prec c} (\llangle c \rrangle - \overline{D_a}) \cup D_a \right).\]

In this example, the characterization of a class a is the set of properties which are defined locally in a or which are (recursively) inherited from its superclasses. This inheritance mechanism allows for exceptions because local definitions that contradict inherited properties override these ones.

The second line of the definition is needed to exclude complementary pairs from the characterization. These could arise in a situation of inconsistency in the class definition or of multiple inheritance from incompatible classes. In such a case the system should not commit itself to any of the possibilities [Tou86].

It may seem strange to allow inconsistent class definitions. However, in HSTs with rules, we want to be able to have simultaneously rules for p and for \(\overline{p}\) and to conclude that there is an ambiguity if both of them are activated. So we allow
inconsistent class definitions, in this example, for the sake of uniformity. It is the burden of the inheritance mechanism to avoid inconsistency in the characterizations because that would mean an empty interpretation function for such classes, as we will see below.

Notice that this construction involves a purely syntactical manipulation of objects in the HST.

The interpretation functions of a pre-interpretation may be any sets of individuals, however nonsensical they may be. We reserve the term interpretation for those cases which are not inherently absurd.

**Definition 11 (Interpretation)** A pre-interpretation \( I \) is an interpretation of a HST if and only if it respects two basic restrictions:

\[
\begin{align*}
c \in \mathcal{C} & \quad \Rightarrow \quad \tau_I(c) \subseteq \kappa_I(c) \\
p \in \mathcal{P}^+ & \quad \Rightarrow \quad (\pi_I(p) \cap \pi_I(\overline{p})) = \emptyset.
\end{align*}
\]

Typical individuals of a class must belong to it \( (\llcorner c \lrcorner)_i \subseteq [c]_i \); and an individual must not verify a property and its negation because such an interpretation would be contradictory. These restrictions are so strong that we rule out all the pre-interpretations not respecting them.

Individuals either belong or do not belong to a class, but the treatment of properties is two-valued partial or three-valued total. A property \( p \) may hold for an individual \( i \) \( (i \in \pi_I(p)) \), it may not hold \( (i \in \pi_I(\overline{p})) \), or it may be undefined.

Although it may be argued that, at the individual level, a property either holds or does not hold, there is a variety of ontological justifications for the third-value. It may mean ignorance of a value that exists but is not known or that does not exist but may become defined in the future. It is also a way to accommodate paradoxical sentences which are neither true nor false. In knowledge bases, two other situations reinforce this choice. Sometimes, when a property \( p \) makes no sense with respect to a certain class of individuals, reasoning as if those individuals were \( \overline{p} \), like in a two-valued approach with closed world assumption, may lead to wrong conclusions. The third value is then seen as *not applicable*. The composition of knowledge chunks from possibly independent sources, each with its own defaults, may easily produce contradictions. If no good reason exists to favor one of the competitors, then the contradictory property is over-defined. The lack of information that could enable the choice may be likened to the one occurring when the property is undefined.

The basic intuition is to deal with \( p \) and \( \overline{p} \) in a totally symmetric way, as if they were different properties, although they are strongly connected by a sort of built-in integrity constraint forcing them to be disjoint.
Our main problem is to relate properties and classes of individuals. The problem here gets trickier because, even if properties applied to individuals were two-valued, some individuals in the class could be \( p \) and the rest \( \overline{p} \). We will return to this subject later, in particular during the discussion of the feasibility of two-valued approaches.

### 1.2.2 Models

**Definition 12 (Model)** *An interpretation \( M \) is a model of a HST \( \mathcal{H} \), written \( M \in \mathcal{M}(\mathcal{H}) \), if and only if, for any two classes \( a \) and \( b \),*

\[
\begin{align*}
\text{if } a \prec b & \Rightarrow [a]_M \subseteq [b]_M \\
[a]_M - \bigcup_{b \prec a} [b]_M & \subseteq \ll a \gg_M \tag{1.4} \\
\ll a \gg_M & \subseteq [\ll a \gg]_M. \tag{1.5}
\end{align*}
\]

The condition (1.4) forces a model to obey a strict class inclusion policy as expected in a taxonomy.

Then (1.5) states that individuals in class \( a \) which do not belong to any of its subclasses must be typical individuals of \( a \). Thus knowledge is only expressed at the class level. The only way to distinguish between two individuals is by including them in different classes. The reverse does not hold. There may be typical individuals of \( a \) that also belong to a subclass of \( a \).

The condition (1.6) assures that the typical subset of a class really collects individuals for whom all the properties in its characterization hold. In other words, if an individual is an exception with respect to any of the characteristic properties of \( a \) then it can not be one of its typical individuals.

**Example 5 Models of a HST.**

In figure 1.2 we present a HST \( \mathcal{H} \) for a modified version of the *Nixon ambiguity* example, both formally and in a graphical way. The characterizations were computed by the inheritance mechanism of example 4. Notice how property \( \neg c \) in class \( b \) overrides \( c \) in \( a \) because \( b \) is more specific than \( a \), and properties \( p \) and \( \neg p \) neutralize each other in class \( n \) because they originate in \( q \) and \( r \), two classes neither of which is more specific than the other. There are no restrictions on the contents of a HST. Every conceivable structure written in the language defined in section 1.1.2 has a meaning.

In figure 1.3 we show some pre-interpretations for \( \mathcal{H} \). The taxonomic interpretation function \( \tau \) is represented by rounded-corner rectangles, the property function \( \pi \) by straight rectangles, and the typical function is written down (\( I_z \) means
\[ \mathcal{H} = \langle \{a, r, q, b, n, \}, \{c, p\}, \{b \prec r, n \prec r, n \prec q, r \prec a, q \prec a\}, \{ \langle a, \{c\}\rangle, \langle r, \{\neg p\}\rangle, \langle q, \{p\}\rangle, \langle b, \{\neg c\}\rangle, \langle n, \{\}\rangle \} \rangle \]

\begin{align*}
& a : \text{american} \\
& r : \text{republican} \\
& q : \text{quaker} \\
& b : \text{bush} \\
& n : \text{nixon} \\
& p : \text{pacifist} \\
& c : \text{coca-cola-drinker}
\end{align*}

Figure 1.2: Two americans.

Figure 1.3: Four interpretations for the HST two americans.
each of the interpretations). The domain of individuals is \( \mathcal{U}_{tx} = \{1, 2, 3, 4, 5, 6\} \). Pre-interpretations are arbitrary but, as in these four no individual belongs to contradictory properties and typical subsets are really subsets of the corresponding class, all of them are also interpretations. In order to be models they must satisfy definition 12. That is not the case of \( I_1 \). It violates (1.4) because \( b \prec a \) and \( [b]_{I_1} = \{3\} \not\subseteq \{a\}_{I_1} = \{1, 2, 4, 5\} \). It also does not conform to (1.5) because \( [q]_{I_1} - [n]_{I_1} = \{5\} \not\subseteq \{b\}_{I_1} \), neither to (1.6) because \( \ll b \gg_{I_1} = \{3\} \not\subseteq \ll b >\) \( \ll_{I_1} = \{\lnot c\}_{I_1} \cap \{\lnot p\}_{I_1} = \{3\} \cap \{2\} = \emptyset \). Interpretation \( I_2 \) is a model, as well as \( I_3 \) and \( I_4 \). You may notice that \( I_3 \) is not minimal in several ways. Individuals 5 and 6 belong to exactly the same classes and properties so, as far as knowledge conveyed by the HST is concerned, they are indistinguishable and could be represent by just one of them. This is the basic idea behind typical individuals. The typical subset of \( a \) includes individual 4, while only 1 is required. And individual 1 enjoys the property \( p \), which, though not forbidden, is not implied by the HST. It is simply undefined. A similar situation occurs with individual 4 with respect to \( \lnot p \), but for a different reason: class \( n \) (multiply) inherits \( p \) and \( \lnot p \) meaning that some of its typical individuals may be \( p \), others \( \lnot p \), and others undefined.

There is an alternative formulation for the conditions (1.4) and (1.5), but that hides the intuitive taxonomic requirement.

**Proposition 13 (Typical subsets)** In a model \( M \in \mathcal{M}(\mathcal{H}) \), a class is the union of its typical subset with all its direct and indirect subclasses:

\[
[c]_M = \ll c \gg_M \cup \bigcup_{b \prec c} [b]_M.
\]  

(1.7)

**Proof:**

We prove that (1.7) is equivalent to (1.4) and (1.5), which are true in any model, according to definition 12.

(1.7) \( \Rightarrow \) (1.4): (1.7) asserts that \([c]_M\) contains the taxonomic denotations of all its subclasses and so \( b \prec c \Rightarrow [b]_M \subseteq [c]_M \).

(1.7) \( \Rightarrow \) (1.5): If \([c]_M = \ll c \gg_M \cup \bigcup_{b \prec c} [b]_M\) then, using set difference on both sides of the equality, \([c]_M - \bigcup_{b \prec c} [b]_M \subseteq \ll c \gg_M\).

(1.4, 1.5) \( \Rightarrow \) (1.7): From the condition (1.5) it is possible to conclude that \([c]_M - \bigcup_{b \prec c} [b]_M - \ll c \gg_M = \emptyset \). So also \([c]_M \subseteq \bigcup_{b \prec c} [b]_M \cup \ll c \gg_M\). On the other hand \( \bigcup_{b \prec c} [b]_M \cup \ll c \gg_M \subseteq [c]_M\) because, by (1.4), subclasses do not outspread their superclass, and, by definition 11, \( \ll c \gg_M \) is a subset of \([c]_M\). So (1.7) follows.
Proposition 14 (Inherent typicality) Every individual is a typical individual in some class:

\[ i \in [c]_M \Rightarrow \exists b \leq c : i \in \llbracket b \rrbracket_M. \]  \hspace{1cm} (1.8)

Proof:

If the individual \( i \) belongs to the denotation of the class \( c \) then, by (1.7) it is either a typical element of \( c \) or it belongs to the denotation of one of its subclasses. In this second case the argument recurs. As the hierarchy is acyclic and finite a class is eventually reached which has no subclasses. Because \( M \) is a model we may use (1.5) to conclude that \( i \) is typical of that class.

If exceptions are to be expressed, the characterization of a lower class \( a \) may contain a property contradictory with another defined above in the hierarchy, in \( b \). As the individuals of \( [a] \) also belong to \( [b] \), the taxonomic denotation is not able to express this nonmonotonic behavior. It is the typical denotation that establishes the connection between classes and their own characterizations.

The semantics of HST is under-specified. The reason for this is to make room for exceptions. Usually, the definition of model of a program is directly based on the notion of satisfaction of a clause. An interpretation is a model of a set of clauses \( h \leftarrow G \) if and only if whenever it satisfies \( G \) it also satisfies \( h \). The corresponding formulation for HST would be similar to the one presented next.

Definition 15 (Satisfaction without exceptions) An interpretation \( I \) satisfies a class definition \( D \) if and only if

\[ \forall a \forall (p \leftarrow Q) \in \bigcup_{c \leq a} D_c, \llbracket a \rrbracket_I \subseteq [Q]_I \Rightarrow \llbracket a \rrbracket_I \subseteq [p]_I. \]  \hspace{1cm} (1.9)

In the definition 12, of model, the first two conditions are concerned with the hierarchy. The third condition, about properties, should be (1.9). According to it, typical individuals of \( a \) should enjoy all the unconditional properties (facts) stated in the superclasses of \( a \) and also all the conditional properties whose premises are themselves enjoyed by \( a \). This result does not allow for exceptions and so (1.9) embodies a monotonic kind of inheritance, which may easily lead to contradictory properties, specially in the intended case of multiple inheritance.

In order to explore different alternatives for the inheritance mechanism, we decided to relax (1.9) introducing here a level of indirection. The condition in (1.6), \( \llbracket a \rrbracket_M \subseteq [\llbracket a \rrbracket]_M \), abstracts out how typical subsets are constrained by the denotations of properties. It is parametric on the characterizations \( \llbracket a \rrbracket \), which should in some way be derived from the theories of the respective superclasses, according to a more or less elaborated definition.
As an example of what the characterization would look like under the monotonic inheritance implicit in (1.9), we adapt the immediate consequence operator of Logic Programming to each class $a$. Once again this amounts to see property rules as clauses with literals abstracted on the target class.

$$\forall (p \leftarrow Q) \in \bigcup_{a \leq c} D_c, Q \subseteq <a> \Rightarrow p \in <a>$$ \hspace{1cm} (1.10)

Each characterization $<a>$ would be the respective least fixed-point. Combining (1.6) with $<a>$ yields (1.9).

The following chapters are devoted to study two more interesting proposals for the characterization of each class in the HST.

Although in definition 12 its contents is left open (the sole requirement is that the intersection of the extensions of the actual properties contains the typical subset), we feel that a reasonable inheritance mechanism should take into consideration the following criteria, ordered by decreasing priority:

1. **Each characterization $<a>$ must be consistent.** So, a means of avoiding contradictory pairs of properties must be devised. A HST obeying this criterion is called *consistent*.

2. In the case of a conflict between two rules, if one is more specific then it should override the other, otherwise both should be neutralized.

3. **The number of rules satisfied in the sense of (1.10) should be maximized**, i.e., exceptions should be minimized, provided that 1. and 2. are respected. If the body of a rule is contained in $<a>$ and its head $p$ is not contradicted by $<a>$, nor by the head of another rule in the same situation, then $p$ should belong to $<a>$ too. If $X_a \subseteq \bigcup_{a \leq c} D_c$ is the set of satisfied rules then this policy can be seen as $X_a$ globally overriding the remaining rules.

4. **The characterization should be minimal.** In other words, properties in $<a>$ should be supported (the converse of (1.10)).

$$\forall p \in <a>, \exists p \leftarrow Q \in \bigcup_{a \leq c} D_c, Q \subseteq <a>.$$ \hspace{1cm} (1.11)

The purpose of this criterion is to avoid the inclusion in $<a>$ of arbitrary, unjustified properties.

### 1.2.3 Standard models

We can identify three dimensions of variability on models. First they depend on the domain of individuals and on the interpretation function ($\kappa$) for classes. We
call a given \(\kappa\) for a given domain a \textit{configuration}. Given a configuration, several possibilities for the extensions of both the typical subsets and the properties are allowed.

The three conditions in definition 12, for interpretations to be models, can be viewed in a constructive way. The first condition validates a configuration. The second imposes a minimum on the typical subsets. The last builds the minimum extensions of the properties based on the typical subsets.

This construction suggests that among the models of a system we may distinguish a \textit{standard model}, using some minimality criteria. Notice that the empty interpretation, where \(\forall c \kappa_I(c) = \emptyset, \forall p \pi_I(p) = \emptyset\), is always a (trivial) model of any HST. We want standard models to have non-empty classes.

**Definition 16 (Standard model)** Among the models of HST \(\mathcal{H}\), those which are subject to the following constraints are called standard models \((M_\mathcal{H})\).

1. \(\forall c \in C \#([c]_{M_\mathcal{H}} - \bigcup_{b < c} \|b\|_{M_\mathcal{H}}) = 1\) (\(\#\) is the cardinality function).

2. Typical subsets are minimal and distinct.

3. Extensions of properties (positive or negative) are minimal.

This definition of standard models seems reasonable because any class must contain at least one individual in order to be relevant and distinct classes must have distinct denotations. The idea is to represent each distinguishable elementary subset in the hierarchy by a single prototype. The set of individuals in this configuration is a standard domain. This domain relates to other possible domains for interpretations of the HST in the following way. Each standard domain element is the representative of the (possibly empty) subset of individuals which belong to exactly the same classes as he does.

Imposing minimality on the typical subsets reduces each of them to the corresponding prototype. However, this choice has a drawback. Specializations of a class, i.e., subclasses that do not contradict the definition of a superclass but may add some other properties, are not considered as part of the typical subset of the superclass, as intuitively one could expect.

We also want to minimize the explicit information contents about properties, either positive or negative, in the model. The third constraint forbids any property not in the characterization of a class to hold for the corresponding typical individual. This is one of the reasons why the models \(I_3\) and \(I_4\) in figure 1.3 are not standard models.

For an example, see in figure 1.3 the interpretation \(I_2\) that is in fact the standard model of the HST in figure 1.2. Individuals 2, 4, and 5 are eligible as typical of
1.2 Extensional semantics

class \( a \) as they belong to \([a]_{I_2}\) and to \([c]_{I_2}\), but were not included in \(\ll a \gg_{I_2}\), according to the minimality criteria. Individual 3 is an exceptional \(a\) that is \(\neg c\) so it must not belong to \(\ll c \gg_{I_2}\).

Before discussing the existence and uniqueness of standard models we will consider inconsistent HSTs.

**Definition 17 (Consistent)** A HST is consistent if and only if every class has a consistent characterization.

**Theorem 18 (Inconsistency)** If the characterization of \(c\) is contradictory with respect to a property \(p\), then the typical denotation of \(c\) must be empty, in every model.

**Proof:**

If \(p, \neg p \in \ll c \gg\) then, by (1.3) and the definition 11 of interpretation

\[
\ll \ll c \gg \gg_I = ([p]_I \cup [Q]_I)
\]

\[
= [p]_I \cap [\neg p]_I \cap [Q]_I
\]

\[
\subseteq [p]_I \cap [\neg p]_I
\]

\[
= \emptyset.
\]

Then, for every model \(M\) we have, by definition 12, \(\ll \ll c \gg \gg_M \subseteq [\ll c \gg]_M = \emptyset. \)

**Corollary 19 (Nonexistence)** An inconsistent HST does not possess a standard model.

**Proof:**

An inconsistent HST has at least one contradictory class. Each of its models must have an empty typical denotation for that class. None of them can be a standard model, because in it typical denotations are singletons.

Notice that inconsistent HST may have many other models.

**Theorem 20 (Existence and uniqueness)** The standard model of a consistent HST exists and is unique modulo the actual individuals.

**Proof:**

The first constraint in definition 16 imposes a configuration, if we abstract from the names of the individuals. The interpretation attributes just one individual to
each class that is a leaf in the hierarchy. Any other class gets exactly one individual which does not belong to any of its subclasses. All these individuals are distinct and constitute a domain. The interpretation function \( \kappa \) assigns to each class the set of individuals in its subclasses plus its own prototype, conforming to (1.4).

The second constraint is equivalent, for the given configuration, to restrict (1.5) to the equality case. So we get classes with exactly one typical individual, which is not typical of any other class. This settles the function \( \tau \) and respects definition 11.

The third constraint defines a unique function \( \pi \), by setting each property equal to the union of the typical subsets of the classes to whose characterizations it belongs. As the HST is consistent, no prototype will be included in the denotations of two contradictory properties, thus respecting the second half of definition 11. ■

1.2.4 Entailment

In this section we first state when an interpretation satisfies a literal and then we define entailment. Literals in the semantics are expressions of the form \( c:p \) where \( c \) is a class and \( p \) is a property. The notion of satisfaction is presented in a two-valued partial fashion. We could also use a three-valued logic as will be shown in section 1.2.5.

**Definition 21 (Satisfaction)** An interpretation \( I \) satisfies a literal \( c:p \) if and only if the typical individuals of \( c \) have the property \( p \).

\[
I \models c:p \iff \llbracket c \rrbracket_I \subseteq \llbracket p \rrbracket_I.
\] (1.12)

The partiality means that it is possible, for certain classes and properties, that neither \( I \models c:p \) nor \( I \models c:\overline{p} \) holds. Notice that a model may satisfy contradictory literals \( c:p \) and \( c:\overline{p} \) only in the case the typical subset of \( c \) is empty because of the requirement of consistency at the level of individuals (see definition 11).

**Definition 22 (Entailment)** A HST \( \mathcal{H} \) entails a literal \( c:p \) if and only if every model of \( \mathcal{H} \) satisfies \( c:p \).

\[
\mathcal{H} \models c:p \iff \forall M \in \mathcal{M}(\mathcal{H}) \, M \models c:p.
\] (1.13)

Although, most of the time, the HST remains implicit in the exposition, it is explicitly mentioned in the entailment relation \( \models \) to emphasize that the truth of a property with respect to a class is not referred to any general framework like, for example, first-order logic, but is mainly related to the fact that the quantification over interpretations is on models of the specific HST under consideration.
Example 6 True properties of the two americans.

The true literals in the models $I_3$ and $I_4$ of figure 1.3 are the following:

<table>
<thead>
<tr>
<th>$I_3$</th>
<th>$I_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a:c$</td>
<td>$a:c$</td>
</tr>
<tr>
<td></td>
<td>$a:\neg p$</td>
</tr>
<tr>
<td>$r:c$</td>
<td>$r:c$</td>
</tr>
<tr>
<td>$r:\neg p$</td>
<td>$r:\neg p$</td>
</tr>
<tr>
<td>$q:c$</td>
<td>$q:c$</td>
</tr>
<tr>
<td>$q:p$</td>
<td>$q:p$</td>
</tr>
<tr>
<td>$b:\neg c$</td>
<td>$b:\neg c$</td>
</tr>
<tr>
<td>$b:\neg p$</td>
<td>$b:\neg p$</td>
</tr>
<tr>
<td>$n:c$</td>
<td>$n:c$</td>
</tr>
<tr>
<td>$n:\neg p$</td>
<td>$n:p$</td>
</tr>
</tbody>
</table>

Only the literals common to all the models follow from $\mathcal{H}$ and it is possible to verify that these are: \{a: c, r: c, r: \neg p, q: c, q: p, b: \neg c, b: \neg p, n: c\}. \hfill \Box

If the characterization $<c>$ of a class is contradictory then, according to theorem 18, its typical subset $<<c>>_M$ is empty for every model $M$ and so it is always contained in the denotation of every property. We get the usual phenomenon of explosion of the semantics of an inconsistent theory ($\forall p, \mathcal{H} \models c: p$), like in classical logic or in the answer-sets formalism [GL90] for logic programming with explicit negation. It is, however, restricted to the inconsistent class and does not spread to the whole HST. It is the responsibility of the inheritance mechanism to avoid contradictory characterizations.

The definition of entailment coincides with what is intuitively expected from the common notion of inheritance as expressed in the next theorem. It stresses the importance of the characterization by establishing a relationship between this syntactic entity and the entailment of certain formulas.

Lemma 23 (Common property) The property $p$ belongs to the characterization of the class $c$ if and only if each typical individual of $c$ enjoys $p$ in every model of the HST $\mathcal{H}$,

$$\forall M \in \mathcal{M}(\mathcal{H}) \quad <<c>>_M \subseteq [p]_M \iff p \in <<c>>.$$  \hfill (1.14)

Proof:

$\Leftarrow$ If $p \in <<c>>$ then, by (1.3)

$$<<c>>_M = [[p \cup Q]]_M = [p]_M \cap [Q]_M \subseteq [p]_M.$$
Then, for every model we have, by definition 12
\[ \forall M \in \mathcal{M}(\mathcal{H}) \quad \llangle c \rrangle_M \subseteq [\llangle c \rrangle]_M \subseteq [p]_M. \]

⇒ This part of the proof is by contradiction. Let us suppose that \( p \notin \llangle c \rrangle \). Then we build an interpretation \( M \) for \( \mathcal{H} \) whose domain has a single individual \( \mathcal{U}_M = \{ i \} \):

\[ \forall a \in C \quad [a]_M = \begin{cases} \{ i \} & \text{if } c \leq a \\ \emptyset & \text{otherwise} \end{cases} \]

\[ \forall a \in C \quad \llangle a \rrangle_M = \begin{cases} \{ i \} & \text{if } c = a \\ \emptyset & \text{otherwise} \end{cases} \]

\[ \forall r \in \mathcal{P} \quad [r]_M = \begin{cases} \{ i \} & \text{if } r \in \llangle c \rrangle \\ \emptyset & \text{otherwise}. \end{cases} \]

As \( \mathcal{H} \) is consistent then \( \llangle c \rrangle \) is not contradictory, i.e., \( r \in \llangle c \rrangle \Rightarrow r \notin \llangle c \rrangle \). So, \( \forall r \in \mathcal{P}^+, [r]_M \cap [r]_M = \emptyset \) and by definition 11 \( M \) is an interpretation. It is easy to verify that \( M \) is a model of \( \mathcal{H} \). As we are assuming that \( p \notin \llangle c \rrangle \) then \( [p]_M = \emptyset \). As \( \llangle c \rrangle_M = \{ i \} \subseteq [p]_M \) we get a contradiction with the premise. So \( p \) must belong to the characterization of \( c \).

This ends the proof.

\[ \blacksquare \]

**Theorem 24 (Characterization)** A consistent HST \( \mathcal{H} \) entails a literal \( c:p \) if and only if the property \( p \) belongs to the characterization of the class \( c \),

\[ \mathcal{H} \models c:p \iff p \in \llangle c \rrangle. \quad (1.15) \]

**Proof:**

By definition, we know that (see (1.13) and (1.12))

\[ \mathcal{H} \models c:p \iff \forall M \in \mathcal{M}(\mathcal{H}) \quad \llangle c \rrangle_M \subseteq [p]_M. \quad (1.16) \]

So, using the lemma 23, the theorem follows.

\[ \blacksquare \]

Theorem 24 shows that the semantics crucially depends on the characterization. It is the set of properties assigned by the inheritance mechanism to each class that shapes the denotations of the objects in the HST, in conjunction with the hierarchic relation. In other words, if a property is in the characterization of a class then all the typical individuals of the class have that property.

Notice that the previous theorem only holds for consistent HST. If a HST contains a contradictory characterization for \( c \) then the quantification mentioned
in (1.13) is done only over models with empty typical subsets for $c$. In virtue of that, the literals $c:p$ are entailed for all $p$, whether $p$ is in the characterization or not. The pre-interpretation $M$ built in the second part of the proof is an interpretation, and a model, only in the case $<c>$ is not contradictory.

Entailment is expensive to test from the definition due to quantification over all models. It would be better if we could devise an easier way to find out which properties hold for each class. This is the main goal behind the construction of the standard model $M_H$.

**Theorem 25 (Standard model)** A literal is entailed by a consistent HST if and only if it is satisfied by its standard model:

$$\mathcal{H} \vDash c:p \iff M_H \models c:p.$$  

**Proof:**

$\Rightarrow$ The *only if* part is immediate from the definition of entailment, because the requirement it states for all models must be true in particular for the standard model.

$\Leftarrow$ The *if* part. In a consistent HST, the standard model exists (see theorem 20). The condition (1.6) imposes that

$$p \in <c> \Rightarrow <c>_M \subseteq [<c>]_{M_H} \subseteq [p]_{M_H}.$$  

The minimization requirements of the standard model (definition 16) force the reverse implication,

$$p \in <c> \iff <c>_M \subseteq [p]_{M_H}.$$  

Because its typical subsets are distinct singletons, the conditions constraining the other classes cannot affect the typical individual of $c$. As its properties are minimal that individual belongs to $[p]_{M_H}$ only if it has to, i.e., if $p \in <c>$. Then, by theorem 24, the result $\mathcal{H} \vDash c:p$ follows.

It is easy to check that the true literals in the standard model $I_2$, figure 1.3, are the entailed literals obtained in example 6.

Most of the time we will be concentrating on a single class at a time in order to find its characterization. So we will sometimes drop the class argument, when the abstraction is unambiguous, and use the designation 'literal' when talking
about property \( p \). In this intensional sense we can see a property as a predicate abstracted on one argument, the class. The connection is established through the characteristic subset \( \ll c \gg \) and not through the taxonomic denotation \([c]\) because we want to allow for exceptions in subclasses of \( c \):

\[
M_{\mathcal{H}} \models c:p \iff \ll c \gg_{M_{\mathcal{H}}} \subseteq [p]_{M_{\mathcal{H}}}
\]

This justifies our initial assignment of two denotations to a class symbol.

Thus we have obtained a perfect match, in the case of consistent HST, between the conclusions derived from the basic definition of entailment and truth in the standard model.

So the proposed semantics for a HST \( \mathcal{H} \) is the set of literals which are true in its unique standard model.

**Definition 26 (Semantics)** The semantics of a consistent HST is

\[
\text{sem}(\mathcal{H}) = \{c:p \mid \mathcal{H} \models c:p\} = \{c:p \mid M_{\mathcal{H}} \models c:p\}.
\]

To extend this definition to every HST, we must change the definition 16 of standard models so that the first condition becomes

\[
\forall c \in C \#([c]_{M_{\mathcal{H}}} \setminus \bigcup_{b < c} [b]_{M_{\mathcal{H}}}) = \begin{cases} 
1 & \text{if } \ll c \gg \text{ is consistent} \\
0 & \text{otherwise.}
\end{cases}
\]

**1.2.5 Models revisited**

What do we mean by saying that knowledge is only expressed at the class level? The building blocks of a HST are just the classes and the properties. So, all the knowledge conveyed by it is bound to refer to, at most, subsets expressible through union, intersection and difference of actual classes. It is impossible to go into further detail, to the level of individuals, for example.

The definition 12 of model is meant to select those interpretations which more closely follow what the HST represents. To talk about the difference between a class and all its subclasses, a sort of smallest distinguishable set in a class, there is the concept of typical subset, which plays a most relevant role in the whole construction.

To say something about the union of two classes, for instance that both enjoy a certain property, the HST may refer to a common superclass. Condition (1.4) naturally ensures that, in a model, the superclass actually contains all the individuals in both classes. If nothing is said to the contrary, a reasonable inheritance mechanism will force the inheritance of that property by the subclasses. But a similar result would be obtained if, instead of using the superclass, the property
were directly included in both class definitions (see figure 1.4; notice that the same interpretation works for both HST, $\mathcal{H}_1$ and $\mathcal{H}_2$).

Intersection is a bit trickier. Syntactically, to state some knowledge about the intersection of $a$ and $b$, there is no alternative to defining a common subclass in order to give a name to the intersection, to which a theory could be attached. Anything said to both classes would pertain to their union and not just to their intersection. Because of this, it seems that a model should not allow the same individual in classes without common subclasses. In fact, we want to smoothly fit the semantics to the concepts and features of the language, i.e., the assignment of individuals to the several objects in the HST should lead to denotations which agree with the capabilities of the language. But, at the semantic level, nothing forbids to build an interpretation with nonempty intersections of the denotations of classes which are, for instance, minimal in the HST, under the $\leq$ ordering (see figure 1.5), and so do not suggest to have any common individuals. Should such an interpretation be a model? Can it be one?

The first impulse is to rule out those unnatural interpretations by adding some extra condition to definition 12. An easy candidate is simply to require that any non-empty intersection of two classes actually is the denotation of an existing common subclass. If a common subclass does not exist then the intersection must be empty.

$$[a]_I \cap [b]_I \neq \emptyset \Rightarrow \exists c : [c]_I = [a]_I \cap [b]_I, c \leq a, c \leq b. \quad (1.17)$$

According to this criterion, the interpretation for the HST $\mathcal{H}_3$ in figure 1.5 is not a model, but the one for $\mathcal{H}_4$ is. Notice that in the former it is possible to reason about sets which are not expressible in the language. For example, if we call $a \cdot b$ to the (nonexistent) class satisfying \(\ll a \cdot b \gg = \{1, 2\}\), it is possible to conclude...
Figure 1.5: From classes to their intersection.

that $I \models a \cdot b: \{p, q\}$, via the semantics, while the programmer has no means to mention the subset $a \cdot b$ in the language. Unfortunately, the interpretation for $\mathcal{H}_5$, which looks like a normal realization, is not a model under (1.17). It would be necessary that either $d$ or $e$ alone were equal to the intersection of $a$ and $b$, instead of just their union. This way we are restricting too much the ability to model concrete situations or, even worse, we are biasing the programmer's choice of language constructs because of limitations in the expressivity of the semantics. We conclude that (1.17) is too strong and propose a weaker version based on the following method of constructing the taxonomic denotations: in a first step, each individual is assigned to a single class, and then they are all propagated to the respective superclasses. More formally:

$$X_i = \{b \mid i \in [b]_I\} \Rightarrow \exists a : a = \min(X_i) \quad (1.18)$$

or, alternatively,

$$[a]_I \cap [b]_I = \bigcup_{c \subseteq a, c \subseteq b} [c]_I. \quad (1.19)$$

Now both interpretations for $\mathcal{H}_4$ and $\mathcal{H}_5$ are models.

We decided not to add (1.18) to the general definition of model because it does not improve the semantics in the relevant situations. The cases in which accepting
an interpretation like the one for $\mathcal{H}_3$ would be really harmful are already ruled out by the very notion of model.

Figure 1.6 represents a pre-interpretation which is not even an interpretation, as $[p] \cap [\bar{p}] = \{1,2\} \neq \emptyset$. Cases like the one in figure 1.5 are diluted by the quantification in the semantics. But the condition (1.18) is significant with respect to the standard model, which is used alone to obtain the semantics. It is necessary to avoid unexpected results due to casual coincidences of typical subsets. This is why they are required to be distinct in the definition 16.

1.2.6 Extending the language

As we mentioned before, the language can be extended to support disjunction and strong negation, besides conjunction, in the bodies of rules or in goals, but keeping the heads of rules as simple properties ($c \in C, p \in \mathcal{P}$).

\begin{align*}
\text{(rule)} & \quad p \leftarrow F \\
\text{(query)} & \quad c : F \\
\text{(formula)} & \quad F ::= p \mid \neg F \mid F \land F \lor F
\end{align*}

(1.20)

1.2.6.1 Constructive semantics

The first step towards giving a meaning to the new formulas is to extend the notion of denotation.

Definition 27 (Denotation, ext.) The denotation of complex formulas in an interpretation is recursively defined up from the denotations of properties previously presented in (1.2).

\begin{align*}
[F \lor G]_I &= [F]_I \cup [G]_I \\
[F \land G]_I &= [F]_I \cap [G]_I \\
[-(F \lor G)]_I &= [-F]_I \cap [-G]_I \\
[-(F \land G)]_I &= [-F]_I \cup [-G]_I \\
[-\neg F]_I &= [F]_I
\end{align*}

(1.21) (1.22) (1.23) (1.24) (1.25)
According to (1.3) sets of properties already have a conjunctive flavor. However, we redefined here the interpretation of the conjunction for the sake of uniformity and to ease the combination with strong negation. The latter is not interpreted directly but only in combination with other connectives. The reason is that, in a three-valued setting, we want to be able to ground all the denotations on the properties, whichever their sign, and to avoid using the complement to the universe.

Notice that we kept the heads of rules as single properties so that it is not possible to state, in the language of the HST, disjunctive conclusions. Defined disjunction represents a different kind of lack of information that we do not address in this work. A consequence of this option is that the characterization is not compelled to encompass the extended formulas but may stay a set of properties as before. So, the theorem 20 on the existence of a standard model remains in force because it only depends on the characterization.

The definitions of satisfaction and entailment need not to be changed except that formulas are acceptable where simple properties were mentioned. It is very convenient to know whether the standard model retains the ability to summarize the whole semantics in the extended language.

**Theorem 28 (Standard model, ext.)** A consistent HST entails a query $c : F$ if and only if it is satisfied by its standard model.

$$\mathcal{H} \models c : F \iff M_\mathcal{H} \models c : F.$$  

**Proof:**

The only if part is immediate from the definition of entailment (1.13), because the requirement it states for all models must be true in particular for the standard model.

The if part is proved by induction on the complexity of the formulas. The base case is for properties, both positive and negative, and it was already proved in theorem 25. The induction step is split by the kind of formula.

$F \lor G$ Due to the specific shape of the standard model, the satisfaction relation can be distributed across disjunction.

$$M_\mathcal{H} \models c : (F \lor G)$$  \hspace{1cm} (1.26)

is equivalent, by (1.12) and definition 27, to

$$\ll c \gg_{M_\mathcal{H}} \subseteq [F \lor G]_{M_\mathcal{H}} = [F]_{M_\mathcal{H}} \cup [G]_{M_\mathcal{H}}.$$ \hspace{1cm} (1.27)

As the typical subset $\ll c \gg$ is a singleton, if it is a subset of the union of two denotations it must also be a subset of at least one of them,

$$\ll c \gg_{M_\mathcal{H}} \subseteq [F]_{M_\mathcal{H}} \text{ or } \ll c \gg_{M_\mathcal{H}} \subseteq [G]_{M_\mathcal{H}}.$$  \hspace{1cm} (1.28)
Again by (1.12), this is equivalent to
\[ M_\mathcal{H} \models c:F \text{ or } M_\mathcal{H} \models c:G. \] (1.29)

By the induction hypothesis, we have
\[ \mathcal{H} \models c:F \text{ or } \mathcal{H} \models c:G \] (1.30)

and, using the definition of entailment (1.13),
\[ \forall M \in \mathcal{M}(\mathcal{H}), M \models c:F \text{ or } M \models c:G \] (1.31)

that implies
\[ \forall M \in \mathcal{M}(\mathcal{H}), M \models c:F \text{ or } M \models c:G. \] (1.32)

From the definition of satisfaction in (1.12), this is equivalent to
\[ \forall M \in \mathcal{M}(\mathcal{H}), \ll M \gg_\mathcal{M} \subseteq [F]_M \cup [G]_M \] (1.33)

that, on its turn, implies
\[ \forall M \in \mathcal{M}(\mathcal{H}), \ll M \gg_\mathcal{M} \subseteq [F \lor G]_M \] (1.34)

by definition 27. Finally, by (1.12), we have
\[ \forall M \in \mathcal{M}(\mathcal{H}), M \models c:(F \lor G) \] (1.35)

and, by (1.13),
\[ \mathcal{H} \models c:(F \lor G). \] (1.36)

\( F \land G \) The proof in this case is considerably simpler because the following four statements are equivalent for all models \( M \), while in the disjunction the proof is restricted to the standard model.
\[ M \models c:(F \land G) \] (1.37)
\[ \ll M \gg_\mathcal{M} \subseteq [F \land G]_M = [F]_M \cap [G]_M \text{ def. 27, (1.12)} \] (1.38)
\[ \ll M \gg_\mathcal{M} \subseteq [F]_M \text{ and } \ll M \gg_\mathcal{M} \subseteq [G]_M \] (1.39)
\[ M \models c:F \text{ and } M \models c:G \] (1.40)

Equation (1.40) is true in particular for the standard model. By the induction hypothesis
\[ \mathcal{H} \models c:F \text{ and } \mathcal{H} \models c:G \] (1.41)
\[ \forall M \in \mathcal{M}(\mathcal{H}), M \models c:F \text{ and } \forall M \in \mathcal{M}(\mathcal{H}), M \models c:G \] (1.42)
\[ \forall M \in \mathcal{M}(\mathcal{H}), M \models c:F \text{ and } M \models c:G. \] (1.43)

Using the equivalence between (1.40) and (1.37)
\[ \forall M \in \mathcal{M}(\mathcal{H}), M \models c:(F \land G) \] (1.44)
\[ \mathcal{H} \models c:(F \land G). \] (1.45)
\neg (F \lor G) \quad \text{According to its denotation, this formula is actually a conjunction. In view of the previous case, the following holds for all models } M \text{ and henceforth for } M_H.

\begin{align*}
M \models c : \neg (F \lor G) \\
\llcorner c \lrcorner M \subseteq [\neg (F \lor G)]_M = [\neg F]_M \cap [\neg G]_M \tag{1.46} \\
M \models c : \neg F \text{ and } M \models c : \neg G \tag{1.12, def. 27} \\
\mathcal{H} \models c : \neg F \text{ and } \mathcal{H} \models c : \neg G \tag{1.48} \\
\mathcal{H} \models c : \neg (F \lor G) \tag{1.49} \\
\text{conjunction} \tag{1.50}
\end{align*}

\neg (F \land G) \quad \text{As before, this formula hides a disjunction. Following the reasoning used in the first case, statements (1.51-1.54) are equivalent while (1.54) implies (1.55).}

\begin{align*}
M_H \models c : \neg (F \land G) \\
\llcorner c \lrcorner M_H \subseteq [\neg (F \land G)]_{M_H} = [\neg F]_{M_H} \cup [\neg G]_{M_H} \tag{1.51} \\
M_H \models c : \neg F \text{ or } M_H \models c : \neg G \tag{1.52} \\
\mathcal{H} \models c : \neg F \text{ or } \mathcal{H} \models c : \neg G \tag{1.53} \\
\mathcal{H} \models c : \neg (F \land G) \tag{1.54} \\
\text{hyp.} \tag{1.55}
\end{align*}

\neg \neg F \quad \text{The proof for this last case is centered on the definition of denotation. We have that, for all models } M,

\begin{align*}
M \models \neg \neg F \tag{1.56} \\
is equivalent to \tag{1.57}
M \models F \\
\text{because } [\neg \neg F]_M = [F]_M \text{ in (27). So, by (1.13), also}
\mathcal{H} \models \neg \neg F \tag{1.58} \\
is equivalent to \tag{1.59}
\mathcal{H} \models F.
\end{align*}

The equivalence is proved.

Corollary 29 (Entailment, ext.) The entailment of complex formulas is compositional and conforms to the following equivalences.

\begin{align*}
\mathcal{H} \models c : (F \lor G) & \iff \mathcal{H} \models c : F \text{ or } \mathcal{H} \models c : G \\
\mathcal{H} \models c : (F \land G) & \iff \mathcal{H} \models c : F \text{ and } \mathcal{H} \models c : G. \\
\mathcal{H} \models c : \neg (F \lor G) & \iff \mathcal{H} \models c : \neg F \text{ and } \mathcal{H} \models c : \neg G \\
\mathcal{H} \models c : \neg (F \land G) & \iff \mathcal{H} \models c : \neg F \text{ or } \mathcal{H} \models c : \neg G \\
\mathcal{H} \models c : \neg \neg F & \iff \mathcal{H} \models c : F. \tag{1.60}
\end{align*}
1.2 Extensional semantics

Proof:

The second, third and fifth lines of (1.60) come directly from the proof of the theorem 28, respectively from the equivalences (1.41)-(1.45), (1.49)-(1.50) and (1.58)-(1.59).

The proof of the first and fourth lines needs an extra step because we have only proved that (1.30) implies (1.36) and (1.54) implies (1.55). However, as by the theorem 28 itself

\[ \mathcal{H} \models c : F \iff M_\mathcal{H} \models c : F \]

and by (1.49)-(1.50) and (1.58)-(1.59)

\[ M_\mathcal{H} \models c : (F \lor G) \iff M_\mathcal{H} \models c : F \text{ or } M_\mathcal{H} \models c : G \]

\[ M_\mathcal{H} \models c : \neg(F \land G) \iff M_\mathcal{H} \models c : \neg F \text{ or } M_\mathcal{H} \models c : \neg G, \]

the equivalences follow.

The equivalences (1.60) show that the entailment of complex formulas is completely determined by the set of literals contained in \( \text{sem}(\mathcal{H}) \). This is why we chose to define the semantics as containing just literals. Such result is a consequence of the extensional semantics we have been building.

Put in another perspective [Wag90], the reason behind this fact is that the logic implicit in (1.60) satisfies two constructiveness principles, with respect to the negation operation:

1. Constructible truth: \( X \models F \lor G \Rightarrow X \models F \text{ or } X \models G \)

2. Constructible falsity: \( X \models \neg(F \land G) \Rightarrow X \models \neg F \text{ or } X \models \neg G \).

In classical logic neither 1. nor 2. hold and so it is possible to conclude the disjunction without concluding any of the disjuncts. The logic of (1.60) insists on being constructive, i.e., on being able to tell which one (or both) of the disjuncts are true. A similar assertion can be made about the constructible falsity. If a conjunction is false then at least one of the conjuncts must also be false.

This logic is more suitable to the kind of programs used in logic programming, as well as in HST, because they do not assert disjunctive facts. The heads of rules are simple properties and so the only way to conclude a disjunction is by concluding one of its disjuncts.

Example 7 Constructiveness.
Let $\mathcal{H}$ be a HST containing a class $c$ and two properties $p$ and $r$. In figure 1.7 we compare two ways of dealing with the disjunction, with respect to four different interpretations. The symbols $p$, $r$, and $c$ label, respectively, the sets $[p]$, $[r]$, and $\langle c \rangle$.

If we try to interpret the connectives directly as operators on properties, the most natural interpretation for the disjunction is the set union. However, this choice is not constructive because, as can be seen in the last row of the table, $\langle c \rangle \subseteq [p] \cup [r]$ but neither $\langle c \rangle \subseteq [p]$ nor $\langle c \rangle \subseteq [r]$.

In the second choice, the class is first distributed across the connectives and then satisfaction is checked against each of the disjuncts. This case is obviously constructive.

The first choice would break the correspondence between the semantics and the standard model because in it the typical subset contains exactly one individual, which is not forced to belong to the extension neither of $p$ nor of $r$ but must belong to their union. The minimization of properties required by the standard model becomes undetermined. It doesn’t help to decide in favor of one property because the standard model would then give more conclusions than the semantics. This is a kind of indefiniteness different from the one considered in HST, that of an individual enjoying neither a property nor its negation.

It is essentially for the same reason that we did not consider, along with the hierarchic relation, assertions like "classes $a$ and $b$ span $c$" because that would force any typical individual of $c$ to belong either to $a$ or to $b$ and, again, to enjoy the disjunction of the respective properties.
Notice that, in the case of conjunction, both interpretations as intersection of property denotations and as a boolean connective lead to the same results: \( \ll c \gg \subseteq [p] \cap [r] \iff \ll c \gg \subseteq [p] \land \ll c \gg \subseteq [r] \). This justifies the conjunctive denotation of a set of properties introduced in (1.3).

\[\square\]

1.2.6.2 Three-valued approach

The definition of satisfaction in (1.12) is two-valued partial. In general, a literal \( c:p \) may be in four different situations with respect to satisfaction in an interpretation \( I \). It may be satisfied; its complement may be satisfied; neither of them; or both. This seems to point to the adequacy of a four-valued denotation for literals. However, for our purposes, the fourth value has little interest. It corresponds to a situation of inconsistency \( (I \models c:p, I \models c: \neg p) \) that can not occur in models except if the typical subset of \( c \) is empty. That is never the case in the distinguished standard model if all the characterizations in the HST are consistent. So, three-valued denotations are enough.

We introduce a three-valued (total) function that is equivalent to the two-valued partial formulation of (1.12), whenever \( I \) is a model and the typical subsets are nonempty. This function ranges over the boolean values \( t, u, f \) standing respectively for true, undefined, and false. The value \( u \) is meant to cater for situations where a given property is not defined in a characterization.

**Definition 30 (Three-valued models)** The valuation of a literal \( c:p \) in an interpretation \( I \), such that \( \ll c \gg_I \neq \emptyset \), is

\[
[c:p]_I = \begin{cases} 
  t, & \text{if } I \models c:p \\
  f, & \text{if } I \models c: \neg p \\
  u, & \text{otherwise.}
\end{cases}
\] (1.61)

The equivalence in theorem 28 between entailment by a HST and satisfaction in its standard model suggests that to find out whether \( \mathcal{H} \) entails \( F \) with respect to a class \( c \) one may concentrate only on the standard model. From the equivalences (1.60) it is possible to conclude that a method to do that is to migrate the negation connective inwards, by successive applications of the last three equivalences, until \( F \) contains only disjunctions, conjunctions and (positive or negative) properties. Simultaneously, the class is distributed across disjunctions and conjunctions. Then the information on the satisfaction by \( M_\mathcal{H} \) of each of the literals thus obtained is recombined to produce the desired result.

As the equivalences (1.60) also hold for \( M_\mathcal{H} \), this is akin to directly evaluating \([c:F]_{M_\mathcal{H}}\). To make the evaluation easier we can extend the denotation of literals
to queries with complex formulas. The following truth tables assign a three-valued interpretation to the connectives conjunction, disjunction and negation.

\[
\begin{array}{c|ccc}
\land & f & u & t \\
\hline
f & f & f & f \\
u & f & u & u \\
t & f & u & t \\
\end{array}
\quad
\begin{array}{c|ccc}
\lor & f & u & t \\
\hline
f & f & u & t \\
u & u & u & u \\
t & t & t & t \\
\end{array}
\quad
\begin{array}{c|c}
\neg & f \\
\hline
f & t \\
u & u \\
t & f \\
\end{array}
\] (1.62)

These define the strong Kleene connectives which operate on truth values. Before applying them to the queries one must first distribute the target class of a formula across the connectives until it applies directly only to properties. This requirement stems from the interpretation in terms of sets of individuals and sheds a new light on the relevance of the constructiveness principles. Alternatively, we may abstract out the class and assign the truth values to the properties and to the formulas, seen as complex properties. Notice that the De Morgan laws are valid with respect to (1.62).

\[-(F \lor G) \equiv \neg F \land \neg G
\quad
-(F \land G) \equiv \neg F \lor \neg G\]

**Theorem 31 (Valuation)** *The standard model satisfies a query if and only if its value is true.*

\[M_\mathcal{H} \models c:F \iff [c:F]_{M_\mathcal{H}} = t\]

**Proof:**

As by theorem 28 \(M_\mathcal{H} \models c:F \iff \mathcal{H} \models c:F\) then the standard model also verifies equivalences analogous to (1.60). The proof of the equivalence with the denotation is by induction on the complexity of the formulas. The base case is for literals and is immediate from the definition 30. The induction step unfolds into several cases.

\[F \lor G \quad \Rightarrow \text{From the equivalences (1.60) } M_\mathcal{H} \models c:F \lor G \iff M_\mathcal{H} \models c:F \text{ or } M_\mathcal{H} \models c:G. \text{ By the hypothesis, } [c:F]_{M_\mathcal{H}} = t \text{ or } [c:G]_{M_\mathcal{H}} = t. \text{ Using the table for the disjunction in (1.62) we may conclude that } [c:F \lor G]_{M_\mathcal{H}} = t.\]

\[\Leftarrow \text{The proof in the reverse direction follows the same path but going backwards.}\]

\[\neg(F \lor G) \quad \Rightarrow \text{For } c:\neg(F \lor G) \text{ to be satisfied by } M_\mathcal{H} \text{ then, by (1.60), both } c:\neg F \text{ and } c:\neg G \text{ must be satisfied too. By the hypothesis, } [c:\neg F]_{M_\mathcal{H}} = t \text{ and } [c:\neg G]_{M_\mathcal{H}} = t. \text{ Using the tables for the conjunction and the negation}\]

\[\quad c:\neg(F \lor G) \Rightarrow \text{For } c:\neg(F \lor G) \text{ to be satisfied by } M_\mathcal{H} \text{ then, by (1.60), both } c:\neg F \text{ and } c:\neg G \text{ must be satisfied too. By the hypothesis, } [c:\neg F]_{M_\mathcal{H}} = t \text{ and } [c:\neg G]_{M_\mathcal{H}} = t. \text{ Using the tables for the conjunction and the negation}\]
in (1.62) we may build the truth table (1.63) for \( \neg F \land \neg G \).

\[
\begin{array}{c|c|ccc}
F & \neg F & f & u & t \\
\hline
G & \neg \lor & f & t & u & t \\
\hline
\neg G & \land & t & u & f \\
\hline
f & t & t & u & f \\
u & u & u & u & f \\
t & f & f & f & f \\
\end{array}
\]

(1.63)

If we apply the negation table to the elements of the table for the disjunction in (1.62) the result coincides with (1.63). So, \([c: \neg (F \lor G)]_{M_H} = t\).

\(\iff\) The proof in the reverse direction starts from the recognition that the only way to get \([c: \neg (F \lor G)]_{M_H} = t\) in (1.62) is by having both \([c: F]_{M_H}\) and \([c: G]_{M_H}\) false. The respective negations must be true and, by the hypothesis, \(M_H \models c: \neg F\) and \(M_H \models c: \neg G\). The equivalences (1.60) lead to the conclusion \(M_H \models c: \neg (F \lor G)\).

\(F \land G, \neg(F \land G)\) These proofs are similar to the previous ones.

\(\neg\neg F\) The proof in this case follows from \(\neg\neg\) being idempotent in the truth table of (1.60).

\[\blacksquare\]

Notice that this result is not generalizable to the other models of \(H\).

### 1.2.6.3 About the rules

The language extension of (1.20) did not include implication in the formulas neither did we give a denotation to the rules. The rules are not classical implications. In Logic Programming they are best seen as inference rules [Wag90, GL90] which produce the respective heads once their bodies become true. But they do not work in the reverse direction. From the falsity of the head one cannot conclude the falsity of the body, i.e., the contrapositive does not hold. In classical logic, from

\[p \leftarrow \neg r\]

\[\neg p\]

(1.64)

one may conclude \(r\) using the contrapositive \(r \leftarrow \neg p\). The implication \(p \leftarrow \neg r\) works in both directions. But from a programmer’s viewpoint, 1.64 expresses a different idea from the classically equivalent fragment

\[r \leftarrow \neg p\]

\[\neg p\]

(1.65)
The latter contains a rule for $r$, so it is natural it may produce $r$, given the truth of its premise. On the contrary, to conclude $r$ from 1.64, one must search the rules' bodies which is operationally inconvenient because it destroys the goal-orientation of the program and is hardly what the programmer intended to convey when writing that fragment.

In HST we also stick to the view that a rule is a conditional property, or conditional fact, for the reasons just pointed out and two more. First, as we are using a partial or three-valued approach, the fact that the head of a rule is false does not necessarily force its body to be false too. Second, one of the fundamental ingredients in HST is the notion of exception, which works by contradiction. So, a conditional fact may be overridden by its opposite (rule with a complementary head) if the latter is more specific than the former. From this one should not conclude that the body of the former is false but only that the rule is not applicable, blocked by overriding.

So, instead of classical negation, disjunction and implication, we have strong negation, constructive disjunction and no implication. This means that $c: (p \lor \neg p)$ is not always true, i.e., the law of the excluded middle does not hold, and that $c: (p \land \neg p)$ is not always false. It may be undefined although it is never true.

1.2.7 Complete information approaches

In an earlier work we tried a complete information approach to HST. There, the denotation of negative properties is the complement of the corresponding positive property: $[\neg p]_t = \mathcal{U}_t - [p]_t$. This means that whether an individual belongs or not to a property is a two-valued total assertion.

It is possible to imagine a fully two-valued system, where also the denotation of literals is total with $c: \neg p \equiv (\ll c \gg_t \in [p]_t)$. However, it seems to be unfair with respect to the syntax of HST, which gives the same status to $p$ and $\neg p$, to define the value of the latter in terms of the former. According to the philosophy of logic programming with the closed world assumption, negative literals represent simultaneously explicit negative information as well as absence of information. So negative properties are reduced to the role of exceptions and defining them in other circumstances becomes rather pointless. In the case of contradictory multiple inheritance it is not possible to stay neutral. Renouncing to assert $p$ means asserting $\neg p$. Insisting on a total semantics, the positive properties are obtained through the quantification over all models. The others are obtained by difference to $\mathcal{P}$.

To enable the system to distinguish ignorance from negative information, we decided to adopt a partial semantics having partial denotations both for properties and for literals. We consider now the implications of keeping the denotations of properties total.
The denotation of literals may be symmetric: \( c : p \equiv \langle \llbracket c \rrbracket \subseteq \llbracket p \rrbracket \rangle, \forall p \in P \). The relationship between a class and a property may be in four different states. The class may have the property; it may have the complementary property; both of them; or none. In the first two cases every model must have the corresponding property. The third case corresponds to a contradiction. Then there are models where the typical individuals have the property, other models where they do not have it and even models where it holds just for some of the typical individuals. The result of the quantification over all models is that neither the property nor its negation are in the semantics. The last case means that the property is undefined. The natural approach is to require that no individual enjoys the property in no model. Due to the two-valued total nature of property denotations this results in the complementary property holding. So a treatment analogous to the case of contradiction would be preferable.

The standard model, as previously defined, does not fulfill its role of summarizing the semantics. If we insist on having complete information about the individuals, the single typical individual of a contradictory class must be decided on every property and the value undefined is never obtained. An alternative could be to assign at least one typical individual to each property inherited, or locally defined, by a class. This requires more than one typical individual in classes where contradictions exist, or relative to which a property is not applicable, by the reason seen above.

We decided to abandon this approach by two main reasons.

- It would force the explicit mention of \( p \) and \( \neg p \) in the typical subsets of the standard model wherever the property is undefined, leading to a more complex model.

- It is possible to delegate the problem of contradiction to the computation of the characterization and remove it from the extensional semantics.

The combination of the standard model of definition 16 with the complete information about individuals leads to the support of some extra negative literals. It is the case that these correspond to a sort of "closed world assumption" [Rei80], in the sense that if it is not stated that a property \( p \) holds for a certain class then we assume that \( \neg p \) does hold. Such a phenomenon does not occur in the semantics because it wipes out everything that is variable, when intersecting all possible models. There is, however, a coincidence between the standard model semantics and the intersection of all models of a related HST \( \mathcal{H}_c \), obtained from the original \( \mathcal{H} \) through an operation called closure.

**Definition 32 (Closure)** The closed version \( \mathcal{H}_c \) of a HST \( \mathcal{H} \) is obtained by setting each local definition in \( \mathcal{H}_c \) equal to the corresponding characterization in \( \mathcal{H} \),
completed with the negative versions of those properties which are not mentioned in it:

\[ \mathcal{D}_a = \langle a \rangle \cup (\mathcal{P}^- \setminus \langle a \rangle). \]  

(1.66)

The closed HST possesses a saturated semantics \((c \in \mathcal{C}, p \in \mathcal{P} \Rightarrow \mathcal{H} \models c: p \text{ or } \mathcal{H} \models c: \neg p)\), which is also a two-valued total semantics, equivalent to the one obtained via the standard model.

Example 8 Properties entailed by a closed HST.

To close the HST \(\mathcal{H}\) in the example two americans, it is enough to add the property \(\neg p\) to the definitions of classes \(a\) and \(n\). Interpretation \(I_c\) in figure 1.8 is the two-valued standard model of the closed HST \(\mathcal{H}_c\). Only positive properties are represented because the denotation of the negative ones is obtained via the complement to the domain. The true literals in \(I_c\), which are also true in all the models of \(\mathcal{H}_c\), are: \(\{a: c, a: \neg p, r: c, r: \neg p, q: c, q: p, b: \neg c, b: \neg p, n: c, n: \neg p\}\).

Notice that the closure operation may affect the typical subsets. If we want to preserve the typical individuals in the standard model under some kind of closure operation which saturates the model, it is necessary to minimize the extension of the typical subsets against the more intuitive choice of maximizing it. In general, a closure tends to reduce the number of individuals which qualify as typical of a superclass because it can add some new properties to the previous characterizations, raising incompatibilities with subclasses that used to be simple specializations.

We consider more adequate to the syntax of explicit negation a symmetric treatment of positive and negative properties and we find convenient to have equivalence between the direct semantics of \(\mathcal{H}\) and its standard model. So, we decided to take the incomplete information view about individuals previously presented in this
chapter. This semantics can distinguish whether a certain property is false or is just undefined. The standard model by itself summarizes all the semantics.

We have presented \( \text{sem}(\mathcal{H}) \) in definition 26 as a two-valued partial semantics. It is equivalent to a three-valued semantics, for consistent HST, in the following way \((p \in \mathcal{P}^+)\):

\[
\mathcal{H} \models_3 c : p = \begin{cases} 
  t & \text{if } c : p \in \text{sem}(\mathcal{H}) \\
  f & \text{if } c : \neg p \in \text{sem}(\mathcal{H}) \\
  u & \text{if } c : p, c : \neg p \notin \text{sem}(\mathcal{H}).
\end{cases} \tag{1.67}
\]

Saturated semantics may of course still occur, meaning that there are no ambiguities left or lack of information.

The partiality in the semantics is genuine in the sense that it is always possible to extend it to a total semantics by making the appropriate choices about each of the properties that are undefined in the characterization. The significance of those choices may be evaluated against the inheritance mechanism behind the characterization.

As a conclusion, we made an effort to attach semantics to the broadest possible set of HST and we succeeded in doing that for them all. Even HST seeming to contain contradictions may be rescued at two different levels. In the process of building the characterizations, a policy may be used that eliminates the contradiction using, for instance, overriding or neutralization, and rendering the rest of the HST's information contents available. If a characterization ends up being contradictory, then every property holds for the corresponding class, but the HST still has a semantics and, in particular, the other classes remain unaffected.
Chapter 2

Bottom-up construction of the characterization

In the previous chapter we left as an open issue the way to obtain the characterizations of the classes in the HST, i.e., the lists of properties holding for them. This corresponds to making explicit the knowledge which is encoded in the theories associated to the classes. A theory is a set of rules implicitly defining the properties in the heads, on the condition that the properties in the corresponding bodies also hold. If a rule body is empty then its head is unconditional and is called a fact. A way to extract the implicit knowledge is precisely to take the facts as a starting base and then iteratively proceed by adding to it the heads of those rules whose bodies are already in the set, until a fixed-point is reached. This is the basic idea of a bottom-up approach to the computation of the set of consequences of a HST. Of course, the whole picture gets more complicated by the consideration of the hierarchic structure relating the theories, which introduces a second iterative dimension.

We will devote the following sections to this subject and leave to the subsequent chapters the study of a goal-directed, or top-down, inference procedure and of a model-theoretic analysis of the HST.

2.1 Predicate inheritance

The goal of this chapter is to state what the contents of the characterization of each class should be. As the starting point for this is the set of theories in the HST, the above mentioned parallel between properties and literals, and also rules and clauses, naturally leads to relate characterizations and interpretations/models of the corresponding theories.
Definition 33 (Interpretation) An interpretation associates a set of properties to a class, 

\[ I : C \rightarrow \mathcal{P}(P). \]

Interpretations in this sense are not to be confused with the notion of interpretation of a HST, as presented in the extensional semantics (cf. chapter 1). Although, from the viewpoint of this semantics, the characterization is seen as a syntactic object, the corresponding notion of interpretation we are introducing has a more semantic character, but in a sort of subsidiary or second level semantics, closer in style to the Herbrand semantics.

A positive property \( p \) in an interpretation \( I^c \) means that \( p \) is true of class \( c \) \( ([p]_I = t) \); a negative property \( \neg p \) means \( p \) is explicitly false \( (f) \); if neither \( p \) nor \( \neg p \) belong to \( I^c \) then \( p \) is undefined \( (u) \). This is the default case. Minimizing an interpretation in the set inclusion sense thus amounts to minimizing explicit information, maximizing the set of undefined literals — this is the three-valued counterpart of the intersection of models in two-valued approaches.

The first inheritance mechanism we developed [DP91] builds on the idea that the set of properties holding for a class \( c \) includes the properties inherited from its superclasses which are not overridden, plus the consequences of the rules in \( D_c \), the class definition for \( c \). The resulting characterization \( <c> \) is then passed to the subclasses of \( c \) as a set of defaults, some of which may then be themselves overridden.

Definition 34 (Defaults) The set of inherited properties, or defaults, \( \Delta^c \) of a class \( c \) is the consistent union of all the characterizations of its superclasses,

\[ \Delta^c = \Theta \bigcup_{c < a} <a> \]. \hspace{1cm} (2.1) \]

Notice that contradictory pairs in the superclasses are neutralized instead of being inherited.

To define the inheritance mechanism, a variant of the immediate consequence operator is used which takes into account the defaults \( \Delta \) (\( I \) is an interpretation):

\[ TP(\Delta)(I) = \{ p \mid p \leftarrow Q \in P, Q \subseteq \Delta \cup I \}. \hspace{1cm} (2.2) \]

The heads of rules are properties and so \( q \) and \( \overline{q} \) may both be generated by \( TP(\Delta) \). Although closely related and interpreted as the negation of each other, they are considered different properties, in the context of the application of the operator. This is akin to the renaming of negative literals as new atoms in [GL90]. From this viewpoint, \( P \) is a positive program and so the operator of (2.2) is the usual immediate consequence operator \( TP \) associated with a logic program \( P \) plus
the facts in $\Delta$. From it can be concluded that the operator $T_P(\Delta)$ is monotonic and continuous, for a given $\Delta$, and so it has a least fixed-point obtainable in $w$ steps,

$$T_P^w(\Delta) = T_P(\Delta) \uparrow w.$$  \hfill (2.3)

The fixed-point $T_P^w(\Delta)$ in (2.3) will be used as an operator which takes a set of properties $\Delta$ and generates all the consequences of the rules in program $P$. If we take for $P$ the definition of a class $a$ then we see the theory $D_a$ as an open definition, a function from sets of properties into sets of properties.

**Definition 35** (Scenario) *A scenario is an interpretation that contains both a set of defaults $\Delta$ and the consequences it induces, $S^a(\Delta) = T_{D_a}^w(\Delta) \cup \Delta$.  

**Lemma 36** (Least fixed-point) *The scenario $T_P(\Delta) \cup \Delta$ coincides with the least fixed-point $T_{P(\Delta)} \uparrow w$ of the common immediate consequence operator associated with the same program $P$ extended with the facts corresponding to the defaults $\Delta$.  

**Proof:**

Let $I^k = T_P^w(\emptyset)$, with $I^0 = \emptyset$, and $J^k = T_P^w(\Delta)(\emptyset) \cup \Delta$, with $J^0 = \Delta$, be the interpretations resulting from $k$ applications of the, resp. common and new, immediate consequence operators to the empty interpretation. As $\Delta$ can be seen as a set of facts added to the program or a set of consequences directly included in the result of $T_P$,

$$I^{k+1} = T_{P \cup \Delta}(I^k) = T_P(I^k) \cup \Delta.$$ \hfill (2.4)

According to (2.2) and noticing that $\Delta \subseteq J^k$ for all $k$, we may write

$$J^{k+1} = T_P(\Delta)(J^k) \cup \Delta = T_P(\Delta \cup J^k) \cup \Delta = T_P(J^k) \cup \Delta.$$ \hfill (2.5)

We will now prove, by induction on $k$, that

$$I^k \subseteq J^k \subseteq I^{k+1}.$$ \hfill (2.6)

**Base case:** $I^0 = \emptyset \subseteq \Delta = J^0$.

**Induction step:** if $I^k \subseteq J^k$ then, as $T_P$ is monotonic, $T_P(I^k) \subseteq T_P(J^k)$ and, using (2.4) and (2.5), $I^{k+1} \subseteq J^{k+1}$, for $k = 0, 1, \ldots$

**Base case:** $J^0 = \Delta \subseteq \Delta \cup T_P(\emptyset) = I^1$.

**Induction step:** if $J^{k-1} \subseteq I^k$ then, again by the monotonicity of $T_P$, $T_P(J^{k-1}) \subseteq T_P(I^k)$ and, using (2.4) and (2.5), $J^k \subseteq I^{k+1}$, for $k = 1, 2, \ldots$

As $T_{P \cup \Delta}$ is continuous, it has a least fixed-point obtainable in $w$ steps. Let us suppose that $I^k$ is that fixed-point, for a certain order $k$. Then $I^k = I^{k+1}$ and so $I^k \subseteq J^k \subseteq I^{k+1} = I^k$ forces $J^k = I^k = J^{k+1}$ to be a fixed-point of $T_P(\Delta) \cup \Delta$. \hfill ■
Due to the presence of negative properties in rule heads, a scenario $S^a(\Delta)$ may be contradictory. If a consequence in $T^a_{\Delta}(\Delta)$ contradicts one of the defaults in $\Delta$, the latter must be overridden. If a pair of contradictory consequences is obtained, a minimal set of defaults supporting them must be overridden. Provided that the class definition is consistent, i.e., does not imply a contradiction by itself ($T^a_{\Delta}(\emptyset) \cap T^a_{\Delta}(\emptyset) = \emptyset$), a suitable set of defaults can be found. So, a class definition may contain rules for both $p$ and $\overline{p}$ and remain consistent as long as $p$ and $\overline{p}$ are not both generated by $T^a_{\Delta}(\emptyset)$.

Predicate inheritance attaches meaning only to the class of consistent HSTs, those containing class definitions which are all consistent.

Example 9 Characterizations through predicate inheritance.

The computation of the characterizations in figure 2.1 (adapted from [San86]) starts with the more general classes and proceeds towards the more specific ones, non-monotonically enriching the sets of conclusions. For instance, mammals are land-dwellers by default inherited from animal:

$$<\text{mammal}> = S^{\text{mammal}}(\Delta^{\text{mammal}}) = \{\text{land-dweller}, \text{air}\} = \Delta^{\text{whale}}.$$  

Whales are an exception explicitly stated with the fact $\neg\text{land-dweller}$. The scenario $S^{\text{whale}}(\Delta^{\text{whale}}) = \{\neg\text{land-dweller}, \text{land-dweller, air}\}$ is inconsistent because the local theory contradicts land-dwellers, in the inherited defaults, forcing its overriding and restricting $\Delta^{\text{whale}}$ to the subset $\Delta' = \{\text{air}\}$. At last,

$$<\text{whale}> = S^{\text{whale}}(\Delta') = \{\neg\text{land-dweller}, \text{air}\}.$$  

Notice that, as air is not overridden by whale, it is inherited from mammal, despite the fact that its support became false.
2.1.1 Models

As seen in example 9, the consequences operator in (2.3) is not guaranteed to give a result consistent with the respective defaults. This may lead to restrict inheritance to smaller sets of defaults. We need a criterion to classify the various possible outcomes. The idea is to choose only those scenarios $S^c(\Delta)$ which are non-contradictory and satisfy all the rules in the class definition.

There are several understandings for the interpretation of the connective $\leftarrow$ in a three-valued setting, depending on the ontological assumptions behind the third value. Some of them, like Luckasiewicz’s and Kleene’s proposals [Tur84], are not adequate to our purpose of defining rule satisfiability because the connective itself is three-valued and that is incompatible with the view of $\leftarrow$ as an inference rule. We want to be definite about whether a scenario satisfies or not every rule in the theory.

Based on similar considerations, Przymusinsky proposed [Prz89] the following. The boolean values are ordered: $t > u > f$. Negation and conjunction are as defined in (1.62). Remember that sets are interpreted conjunctively. A rule is satisfied by an interpretation $I^c$ if the head is greater than the body:

$$[p \leftarrow Q]_{I^c} = [p]_{I^c} \geq [Q]_{I^c}. \tag{2.7}$$

\[
\begin{array}{l|ccc}
\leftarrow & f & u & t \\
\hline
f & t & f & f \\
u & t & t & f \\
t & t & t & t \\
\end{array}
\tag{2.8}
\]

The valuation of a rule is always $t$ or $f$. As [Prz89] points out, this is essential to Logic Programming in order to make it easier to find models and to keep the constructive side of rules in force.

Example 10 Hidden inconsistency.

Suppose that the definition for a class $c$ is $D_c = \{\neg p \leftarrow r\}$ and that its set of defaults is $\Delta^c = \{p\}$. As the rule is not active, the set of consequences $T_d^c(\Delta^c)$ is empty reducing $S^c(\Delta^c)$ to $\{p\}$. This scenario is non-contradictory but it does not satisfy the rule because $f \leftarrow u$ is false, according to (2.8). So, $p$ must be ruled out, although $\neg p$ does not belong to the scenario. The inconsistency is hidden in the truth-table. The empty scenario $S^c(\emptyset) = \emptyset$ is a model as it satisfies all the rules.

There is another model of $D_c$ including $p$ which is $\{p, \neg r\}$, but to find it out we must resort to the contrapositive $\neg r \leftarrow p$, against the usual practice in Logic Programming of considering the rules unidirectional. This amounts to switch $r$, a literal in the body of a rule, from undefined to false. $\Box$
There is a different formulation for the classical notion of model, which is also expressed by the same truth table (2.8), but helps to clarify example 10. An interpretation $I^c$ classically satisfies a rule $p \leftarrow q_1, \ldots, q_n$ if and only if $(q_1, \ldots, q_n \subseteq I^c \Rightarrow p \in I^c) \land (\bar{p} \in I^c \Rightarrow \bar{q}_1 \lor \ldots \lor \bar{q}_n \in I^c)$. An interpretation $I^c$ is a classical model for $c$ if and only if it is non-contradictory and classically satisfies all the rules in the class definition $D_c$.

The first half of the conjunction requires that, in the truth-table (2.8) for the connective arrow, $(u \leftarrow t) = f$ and $(f \leftarrow t) = f$, while the second half, the contrapositive, imposes, again, $(f \leftarrow t) = f$ and also $(f \leftarrow u) = f$. It is precisely this last element that requires an implicit mechanism of side-effects, as suggested in example 10. If one adds to the empty model the literal $p$, then $\neg r$ must also be added. To avoid this side-effect we propose to weaken the classical notion of model\footnote{Formulation suggested by Luís Monteiro in July 1991 where the unidirectional model was called semimodel.} by dropping the second half of the conjunction.

**Definition 37 (Pre-model)** An interpretation $I^c$ satisfies a rule $p \leftarrow q_1, \ldots, q_n$ if and only if $(q_1, \ldots, q_n \subseteq I^c \Rightarrow p \in I^c)$. An interpretation $I^c$ is a pre-model for $c$ if and only if it satisfies all the rules in the class definition $D_c$.

**Definition 38 (Model)** A consistent pre-model is a model.

The truth-table of the arrow connective under this notion of unidirectional model coincides with the one for the external Bochvar connective.

\[
\begin{array}{ccc}
\leftarrow & f & u & t \\
f & t & t & f \\
u & t & t & f \\
t & t & t & t \\
\end{array}
\]

(2.9)

It diverges from (2.8) just in the framed element. Notice that this distinction is meaningful only in a three-valued setting. In the two-valued case, classical models and models coincide. Now, $S^c(\{p\}) = \{p\}$ is a model of $D_c$, in example 10, and this can be concluded just by the application of the operator $T^c_{D_c}(\{p\})$ and the verification that the resulting scenario contains no contradictory pairs of properties. There is no need to go through the rules with false heads to check if their bodies are false, too. A rule with a false or undefined body is just considered inactive, independently of its head value, and so cannot be falsified.

On the contrary, active rules with true bodies may be falsified in two ways: if the head is either false or undefined. In the first case, one gets $f \leftarrow t$ which is false even in the weakest version of three-valued logic, unless the rule itself is
overridden. But, if that were the case, it would be removed from the set of rules to be satisfied by the interpretation. The second case, \( u \leftarrow t \), which is also false in all the common three-valued logics, is relevant in sheer neutralizations of two contradictory rules, which happen to be simultaneously active. Having taken care of the previous case by removing the explicitly overridden default, we may in this case want to search in the bodies the defaults responsible for the contradiction and remove them too. The neutralization solution, i.e., canceling both contradictory properties after they have been produced by \( T^*_D(\Delta^c) \), may be unpleasant because it would leave the guilty defaults in the scenario and force a further relaxation of the very notion of model, namely that \( (u \leftarrow t) = t \).

**Lemma 39 (Model identity)** An interpretation is a (three-valued) pre-model of a program \( P \) if and only if it is a (two-valued) classical model of the program \( P^* \) obtained from \( P \) by renaming the negative properties as new properties.

**Proof:**

The truth table (2.10) is the projection of (2.9) that collapses the values \( f \) and \( u \) into a single value.

\[
\begin{array}{c|ccc}
\rightarrow & f \text{ or } u & t \\
\hline
f \text{ or } u & t & f \\
t & t & t
\end{array}
\]

If the interpretation is consistent the coincidence between the two notions of model follow from the coincidence of (2.10) with the truth table for the classical notion of satisfaction. This shows that, from the point of view of satisfaction, \( f \) and \( u \) behave the same.

If the interpretation is inconsistent then both contradictory properties are seen as true by classical satisfaction. Definition 37 has a similar effect because it just requires the presence of the properties in the interpretation, irrespective of its value.

We have already highlighted the similarity between (2.3) and \( T_P \) in lemma 36. The difference lies on the role of the set \( \Delta \). It is like having some previously generated conclusions thrown into every scenario.

Having in mind that, in the context of three-valued semantics, several orders are commonly used, we precise that the order we have always implicit is the information or Fitting order, which corresponds, in the representation based on single sets of positive and negative literals that we are using, to set inclusion.

**Theorem 40 (Least pre-model)** The scenario \( S^c(\Delta) \) is the least pre-model of \( D_c \cup \Delta \).
Proof:

It is known [Llo87] that the least fixed-point of $T_P$ is the least model of a positive program $P$. If the program is a class definition extended with a set of defaults, we conclude, using the trick of renaming the negative properties as new properties, that $T_{D_c \cup \Delta} \uparrow w$ is the least classical model of $(D_c \cup \Delta)^*$. As, by lemma 36, $T_{D_c \cup \Delta} \uparrow w$ is equal (modulo the renaming) to $T_{D_c}^c(\Delta) \cup \Delta = S^c(\Delta)$, we may conclude that the latter is the least classical model of $(D_c \cup \Delta)^*$ and also, by lemma 39, the least pre-model of $D_c \cup \Delta$.

Notice that if a rule is not active for the two-valued fixed-point operator, it is not active for the three-valued operator either. In both cases we use only the positive form of the rules and ignore the contrapositive.

Corollary 41 (Inconsistent scenarios) If a scenario $S^c(\Delta)$ is inconsistent then there is no model for the class definition $D_c$ with the given defaults $\Delta$.

Proof:

As $S^c(\Delta)$ is the least pre-model, every other pre-model of $D_c \cup \Delta$ will be a superset of the former and thus inconsistent, too.

At first sight, if a class definition contains contradictory rules simultaneously active, then it seems that knowledge is ill-represented in the HST. However, as explained before, we want to be able to extract information from the rest of the system, instead of a priori ruling it out. First we will study the HSTs without that problem, and then we will see how to deal with it.

2.1.2 The basic overriding situation

In a first attempt, the characterization could be chosen to be the least model, irrespective of the defaults. There is always one if the class definition is consistent. But this amounts to ignore inheritance, i.e., to set $\Delta = \emptyset$ in the scenario, which becomes $S^c(\Delta) = T_{D_c}^c(\emptyset)$. In general, we would like to include in the characterization as much as possible of the inherited properties.

This goal is hindered by the occurrence of contradictions, in particular between the consequences of the theory and the defaults. As the theory is more specific the contradicted defaults must be overridden, in some way. To be able to handle separately the defaults and the consequences of a theory, we refine the notion of scenario to split its two components.

Definition 42 (Split scenario) A split scenario is a pair $\langle \Delta, \Gamma_c(\Delta) \rangle$ where $\Delta$ is the set of inherited defaults and $\Gamma_c(\Delta) = T_{D_c}^c(\Delta)$ is the set of its consequences through the theory $D_c$. 
2.1 Predicate inheritance

Abusing language, we will sometimes identify a split scenario \( \langle \Delta, \Gamma_c(\Delta) \rangle \) and the corresponding scenario \( S^c(\Delta) \) and talk, for example, about a property belonging to a split scenario or about a split scenario as being a model.

To ease the comparison of the several possibilities and to get a deeper understanding of the overriding process by simplification of defaults, we introduce a lattice of split scenarios \( \mathcal{L} \) associated to a class \( c \) which inherits the set of properties \( \Delta^c \). Each element \( \langle \Delta, \Gamma_c(\Delta) \rangle \) of \( \mathcal{L} \) is a split scenario where \( \Delta \subseteq \Delta^c \) is a subset of the defaults for \( c \). \( \mathcal{L} \) is ordered by set inclusion among the defaults \( \Delta \). So, the top element is \( S^c(\Delta^c) \), the scenario based on the whole set of inherited defaults, and the bottom element is \( \langle \emptyset, \Gamma_c(\emptyset) \rangle \), fully ignoring the defaults. If \( n \) is the number of defaults in \( \Delta^c \), the number of elements in the lattice is \( 2^n \), the number of subsets of \( \Delta^c \).

Intuitively, the characterization should be the greatest model, if there is one, or a composition of the alternative maximal models that can be found in the lattice.

**Example 11 Simple overriding.**

\[
\begin{align*}
\Delta^c &= \{ p, r \} \quad \text{(inherited properties)} \\
\mathcal{D}_c &= \{ \neg r \leftarrow p \} \quad \text{(local definition)} \\
<\varepsilon> &= ? \quad \text{(characterization of } c) \\
\end{align*}
\]

![Figure 2.2: Lattice of split scenarios.](image)

With a maximum of two defaults, the lattice in figure 2.2 has four elements. The subset \( \Delta \) of the actual defaults in a lattice element is signalized by a box. \( S_1 \) contains a contradictory pair \( (r, \neg r) \), so it clearly is not a model. Models in the lattice are distinguished by a double circle. If we let the rule in \( \mathcal{D}_c \) override \( r \), we get \( S_3 \), which is non-contradictory and thus is a model. Scenario \( S_4 \) is subsumed by \( S_3 \). However, \( S_2 \), which is not subsumed by \( S_3 \), is a model, too. It corresponds to overriding \( p \), instead of \( r \), thus removing the support for the contradiction.

Between these two alternative models we intuitively prefer \( S_3 \), because it is closer to the notion of direct overriding. From a contradiction between the conclusion of a rule in the local theory and an inherited default, we decide to override...
the latter. Starting with $S_1$, to recognize that going to $S_2$ eliminates the problematic contradiction on $r$, involves looking into the bodies of the rules for $\neg r$ in $D_c$ in search of a suitable set of defaults which, if overridden, would render all those rules inactive, i.e., one must use the contrapositives.

The answer is then $\langle c \rangle = S^c(\{p\}) = \{p, \neg r\}$. □

**Theorem 43 (Lattice)** All the scenarios below a model in $L$ are also models.

**Proof:**

We already know from theorem 40 that whenever $S^c(\Delta)$ is non-contradictory it is the least model of $D_c \cup \Delta$. Let us suppose that the split scenario $\langle \Delta', \Gamma_c(\Delta') \rangle$ is a model. As the operator $\Gamma_c$ is monotonic, a consequence of the monotonicity of the basic immediate consequence operator of (2.2), if $\Delta \subseteq \Delta'$ then $\Gamma_c(\Delta) \subseteq \Gamma_c(\Delta')$ and $\langle \Delta, \Gamma_c(\Delta) \rangle \subseteq \langle \Delta', \Gamma_c(\Delta') \rangle$. As $\langle \Delta', \Gamma_c(\Delta') \rangle$ is consistent, all its subsets are consistent, too. The only point left is to prove that scenarios which are below a model in $L$ satisfy all the rules in $D_c$.

As $\langle \Delta', \Gamma_c(\Delta') \rangle$ is a model, for no rule $p \leftarrow Q$ does it happen that $[Q]_{\langle \Delta', \Gamma_c(\Delta') \rangle} = t$ and $[p]_{\langle \Delta', \Gamma_c(\Delta') \rangle} \in \{f, u\}$. When going down the lattice, the set of defaults $\Delta$ becomes strictly smaller than $\Delta'$. If $\delta \in \Delta' \setminus \Delta$ (one of the withdrawn defaults) belongs to the body of a rule, then its value becomes undefined, or remains unchanged if $\delta \in \Gamma_c(\Delta)$, and in any case the rule is still satisfied. If $\delta$ is the head of a rule and the body is true, then it is generated by $\Gamma_c(\Delta)$ and $\delta$ remains true, otherwise the body is undefined or false and the rule is satisfied. Properties that depend on $\delta$ may be analyzed in the same way. They can never switch signal, but at much become undefined. □

**Corollary 44 (Semi-lattice)** The set of models in $L$ forms a lower semi-lattice ordered by inclusion of the default components.

**Proof:**

A set is a semi-lattice if for any pair of elements, the respective meet also belongs to the set. Let us take two models in $L$, $\langle \Delta', \Gamma_c(\Delta') \rangle$ and $\langle \Delta'', \Gamma_c(\Delta'') \rangle$. The meet $\langle \Delta' \cap \Delta'', \Gamma_c(\Delta' \cap \Delta'') \rangle$ is also a scenario in $L$ and, as it is below each of the considered models, by theorem 43, it is also a model. □

**Example 12** Theorem 43 does not hold for classical models.
\[ \Delta^c = \{ \neg r \} \]
\[ \mathcal{D}_c = \{ \neg q, \quad q \leftarrow r \} \]

\[ S_1 \]
\[ \neg r \]
\[ \neg q \]
\[ S_2 \]
\[ \neg q \]

Figure 2.3: A lower scenario that is not a classical model.

The theory \( \mathcal{D}_c \) in figure 2.3 is rather odd. It is not really inconsistent because \( S_2 \) is not contradictory but, after stating \( \neg q \), the rule \( q \leftarrow r \) is pointless. Nevertheless, rejecting such a program seems too strong, as its meaning is clear.

This example disproves the conjecture that all the scenarios below a classical model are also classical models. Both \( S_1 \) and \( S_2 \) are models but, while \( S_1 \) is a classical model, the same does not happen with \( S_2 \) because the rule \( q \leftarrow r \) evaluates to \( f \leftarrow u \), which is false according to the table (2.8). In \( S_1 \) it is the default \( \neg r \) that fixes \( r \) to false and allows the stronger requirements of the classical satisfaction relation to be fulfilled. \( \square \)

Due to the monotonic character of the consequences operator, the top scenario \( \langle \Delta^c, \Gamma_c(\Delta^c) \rangle \) in the lattice contains all the other elements. The bottom scenario \( \langle \emptyset, \Gamma_c(\emptyset) \rangle \) is the smallest set of properties that we must attach to class \( c \) because, assuming \( \mathcal{D}_c \) is consistent, it is its least model. From theorem 44, it follows that there exists one (or more) maximal models covering the sublattice.

Example 11 showed a situation with two maximal models and also suggested that they were not equally convincing. There is a deep reason for this asymmetry. From \( \{ r, \neg r \leftarrow p \} \) it is possible to conclude \( \neg p \) in classical logic, by using case analysis or through the contrapositive \( \neg p \leftarrow r \). Such conclusion could justify overriding \( p \). But doing this is against the tradition of Logic Programming [GL90] where \( \leftarrow \) is viewed as an inference rule, in a constructive way, and not as a classical implication. Moreover, if the rule has multiple premises as in \( \neg r \leftarrow p, s \) then its contrapositive is \( \neg p \lor \neg s \leftarrow r \), bringing in the problems of classical disjunction. So, the contrapositive does not hold in HST, in the sense that the process of eliminating conflicts between rules and defaults never introduces new information (new literals like \( \neg p \)), but is restricted to remove literals from \( \Delta \). This operation, by itself, just changes properties from \( t \) to \( u \), unless they are also defined independently. From a different perspective, but still for the same reason, we notice that, in figure 2.2, while \( S_2 \) and \( S_3 \) are both models, only \( S_2 \) is a classical model, unless \( \neg p \) is added to \( S_3 \).

To decide if a certain default \( \delta \) is overridden or not, we prefer to look for rules
with \( \delta \) in the head and see if the corresponding body is true. The motivation is to preserve the goal-orientation of the overriding method. The corresponding operation on the lattice is the removal of a contradictory default. If \( \langle \Delta, \Gamma_c(\Delta) \rangle \) is the current scenario and \( \delta \) contradicts \( \Gamma_c(\Delta) \), then a preferred step is to go down the lattice to the scenario \( \langle \Delta - \{\delta\}, \Gamma_c(\Delta - \{\delta\}) \rangle \). A preferred path is a path composed solely of preferred steps. A maximal model is preferred if all the paths between the top and itself are preferred.

**Definition 45 (Predicate inheritance)** In a lattice \( \mathcal{L} \), associated to a class \( c \), with defaults \( \Delta^c \), if some of the maximal models are preferred, then the characterization \( <\top> \) is their meet, otherwise it is the meet of all the maximal models.

The conclusion of the motivating example 11 is a posteriori justified by this definition.

**Example 13** Two common test cases.

\[
\begin{align*}
\Delta^a &= \{q\} \\
\mathcal{D}_a &= \{-q \leftarrow q\}
\end{align*}
\]

\[
\begin{align*}
\Delta^b &= \{p, r\} \\
\mathcal{D}_b &= \{\neg p \leftarrow r, \\
&\quad \quad \quad \quad \quad \quad -r \leftarrow p\}
\end{align*}
\]

![Figure 2.4: The odd and the even loops.](image)

The two situations represented in figure 2.4 are the HST specific version of two limit cases commonly used to test the behavior of nonmonotonic formalisms. The theory \( \mathcal{D}_a \) contains an odd loop, i.e., a situation where a literal depends on its opposite. The scenario \( S_1 \) is clearly not a model. So the only model is \( S_2 \), where \( q \) is undefined.

In the theory \( \mathcal{D}_b \) there is a pair of literals, each depending on the negation of the other, a similar situation to the even loop in logic programs. The problem here is that overriding any of the defaults is enough to obtain a model, which is actually also a classical model. So the lattice has two symmetric maximal models, \( S_4 \) and \( S_5 \), and there is seemingly no good reason to favor one of them. This is
the canonical example behind the choice, in definition 45, of the meet of maximal models as the semantics of a theory, under a skeptical approach. So, $<b>$ is empty, too.

Example 14 Skeptical approach.

\[ \Delta^c = \{p, q, r\} \]
\[ D_c = \{s \leftarrow p, r; \]
\[ s \leftarrow q, r; \]
\[ t \leftarrow r; \]
\[ \neg p \leftarrow q; \]
\[ \neg q \leftarrow p \} \]

Figure 2.5: Meet versus intersection.

In the example of figure 2.5, as in the one of figure 2.4, there are two symmetric maximal models, $S_2$ and $S_3$. A skeptical approach to overcome this ambiguity could be to take every literal common to both models, i.e., the intersection $S_2 \cap S_3 = \{r, s, t\}$. Such solution has the disadvantage that it is not in general a scenario. There is no rule supporting the property $s$ even if it belongs to both models. To avoid these floating literals, we opted to define the characterization as being a concrete scenario. So, instead of the intersection of the preferred models

\[ \langle\{q, r\}, \Gamma_c(\{q, r\}) \rangle \cap \langle\{p, r\}, \Gamma_c(\{p, r\}) \rangle, \]

definition 45 picks

\[ \langle\{q, r\} \cap \{p, r\}, \Gamma_c(\{q, r\} \cap \{p, r\}) \rangle = \langle\{r\}, \Gamma_c(\{r\}) \rangle = S_5 = \{r, t\}, \]

their meet.

Example 15 Greatest model not a classical model.
\[ \Delta^c = \{ \neg r, p \} \]
\[ D_c = \{ \neg p, t \leftarrow p, r \leftarrow t, s \leftarrow \neg r \} \]

Figure 2.6: A class with its own facts.

The theory in figure 2.6, as opposed to the previous examples, is able to reach at some conclusions without the support of the inherited defaults. It is the case of \( \neg p \) in the bottom scenario.

The main point we want to stress in this example is that the unique maximal model \( S_2 \) is not a classical model because \( r \) is false and \( t \) is undefined.

Also, one might wonder if the method of choosing always a contradicted default to be eliminated in each step is enough to ensure that, starting from the top, a maximal model is reached. Such conjecture is disproved here. Removing \( \neg r \) in \( S_1 \) gives \( S_3 \), which is not a model. Removing then \( p \) gives \( S_4 \), a model, but not maximal. As a practical hint, one should start the elimination procedure by choosing the defaults which contradict the bottom of the lattice, as its contents are inevitably in any model.

Example 16 Preferred paths.

According to definition 45, the characterization sought for in figure 2.7 coincides with \( S_7 \). Although not covered by \( S_7 \), the model \( S_6 \) is not considered for composition with the former because, among the paths between the top \( S_1 \) and \( S_6 \), there is one which is not composed solely by removals of contradictory defaults. In fact \( S_1 - S_4 - S_6 \) is not a preferred path. The default \( p \), eliminated between \( S_4 \) and \( S_6 \), is not contradictory in \( S_4 \).

2.1.3 Contradictory images

In the previous subsection we examined how the contradictions between the consequences of a theory and the defaults it inherits are resolved by overriding the latter. We will now extend that study to the case where the set of generated consequences itself is contradictory. This situation can happen only if the theory
contains definitions both for $p$ and for $\overline{p}$, a sort of potential inconsistency disclosed by the presence of certain defaults.

Example 17 *Never inheritable property.*

The theory in figure 2.8 is a special case in that a single default $r$ supports the contradiction on $q$. So, $r$ can never be inherited. This conclusion may be reached through case analysis. Fortunately, definition 45, of characterization, is still working and indicates $<c>\geq S_2$, the greatest model. The oddity of the situation is manifest in that $S_2$ is not preferred, i.e., it results from removing from the defaults a literal which is not contradicted. To remove the contradicted $q$ is pointless, as the resulting $S_3$ is equal to the top $S_1$. This illustrates the need for the
proviso made in definition 45 for cases where no maximal model can be obtained only through preferred paths.

The definition of characterization remains applicable in this more general setting because it takes into consideration the whole lattice of scenarios ordered by the default component. A more incremental approach, which was already not clear in the previous case, is now a lot harder because it becomes necessary to look into the rule bodies in search of minimal sets of properties supporting the contradictions. These sets may be not unique. In the presence of equally plausible alternatives, a composition of them can be the intended answer.

Example 18 Potential inconsistency revealed.

\[
\Delta^e = \{p, r, t\} \\
\mathcal{D}_e = \{p \leftarrow r, \neg p \leftarrow t\}
\]

![Diagram of indirect contradiction](image)

Figure 2.9: Indirect contradiction.

Figure 2.9 is another instance of contradictory consequences. This time there are two properties, \(r\) and \(t\), supporting the contradiction, instead of a single one. So, it is possible to obtain a model by the removal of just one of them. This is the case of \(S_4\) and \(S_5\). There seems to be no good reason to prefer one against the other. According to our usual skeptical viewpoint, and to definition 45, we set \(<c>= S_8\), the meet of \(S_4\) and \(S_5\).

Example 18 shows that the meet is the kind of composition used when there are several alternative maximal models, also in the case of contradictory generated consequences. We have already said that it is not easy to define an incremental algorithm which, starting from the top of the lattice, is able to select an appropriate default to override in each step and guarantee that one (or, preferably, all) maximal model is reached. However, we have given some hints along with the examples to
help in this process. As a last suggestion, we introduce next a method to simplify theories and make more evident the supports of contradictions in the generated consequences.

The method presented here is based on unfolding the rule bodies in the theory until they contain only defaults or cycles are found. The unfolding must be exhaustive and produce all the potential combinations of defaults in the bodies. A fact unfolds to true, of course. A literal that is not a default and for which there is no definition unfolds to false. A cycle implies the value undefined for the corresponding head, if it is not generated by other means. So only the rules that unfold completely to defaults are relevant. The result of this program transformation is an explicit function from defaults into all the possible conclusions. Then it is a matter of finding the several minimal sets of defaults to eradicate, and purge them all. The remaining consequences can be obtained directly from the unfolded theory.

Example 19 Induced contradictions in $\Gamma_c(\Delta^c)$.

$$\Delta^c = \{r, s, t, u\} \quad \mathcal{D}_c = \begin{cases} p \leftarrow r, s \\ \neg p \leftarrow t, u \\ t \leftarrow s \\ r \leftarrow t, v \\ v \end{cases}$$

The generated consequences are $\Gamma_c(\Delta^c) = \{v, r, t, p, \neg p\}$, including the contradiction $p \rightarrow \neg p$. The first step to find the characterization is to unfold the bodies in every possible way. The result is

$$\mathcal{D}_c' = \begin{cases} p \leftarrow r, s \\ p \leftarrow t, s \\ p \leftarrow s \\ \neg p \leftarrow t, u \\ \neg p \leftarrow s, u \\ t \leftarrow s \\ r \leftarrow t \\ r \leftarrow s \\ v \end{cases}$$

To take back the contradiction $p \rightarrow \neg p$, one must take back either $p$ or $\neg p$ (or both). To undercut the support of a literal $l$ it is necessary to pick up one default from each rule for $l$ in the unfolded theory, to set them all inactive. The possible alternatives are: $\{r, t, s\}, \{r, s\}, \{s\}, \{t, s\}, \{t, u\}, \{u, s\}, \{u\}$. Among these, only two are minimal, $\{s\}$ and $\{u\}$. So, removing either $s$ or $u$ is enough to ensure
that the above mentioned contradiction no longer exists. As there is no reason
to override one and keep the other, both defaults are overridden. The remaining
conclusions are $\langle\emptyset\rangle = \{v, r, t\}$. 

There is a close relationship between this method and the direct manipulation
of the lattice because to each minimal set of contradiction supporting defaults
corresponds an element in the lattice. The union of those minimal sets corresponds
to the meet of the respective lattice elements.

The mechanism described implements the property flow metaphor [HTT90],
where properties are viewed as percolating through the hierarchy from the most
general to the most specific classes. Each class definition is compiled out with
respect to the inherited properties before passing the updated set to its subclasses.
This kind of inheritance is similar to the definition in [MP92] of predicate inheri-
tance, as opposed to clause inheritance. The main difference to that approach lies
in the way overriding is dealt with. In [MP92] two modes are introduced: extension
and overriding, at the level of a complete predicate definition (set of clauses for the
same predicate name). The two modes are needed because no explicit negation
is allowed in the heads of clauses. Working with positive and negative literals,
it is possible to override only part of a predicate by stating just the contradictory
property(ies). It results in a finer control over the inheritance process. Non
contradictory properties extend the already existing ones for that predicate.

The top-down nature of predicate inheritance reminds the possibility of using
an ask/tell model. The process could start from the maximal elements in the
hierarchy, with an empty blackboard. At each class, the current set would be asked
by the respective theory for the pertinent defaults and the properties concluded
could be incrementally told. The overriding requires a nonmonotonic tell operation,
in which telling a new property implies the removal of its opposite. An appropriate
scheme to ensure neutralization would also be necessary. But such a model is not
satisfactory because it can not be aware of the dependencies expressed by the rules.

The main problem with predicate inheritance is that it allows for a rather lim-
ited expressiveness because after the set of conclusions is computed the functional
character of the rules behind it is lost. Inheriting just (sets of) properties does
not keep track of the dependencies among them. The result may be to retain in
subclasses conclusions whose support has been overridden, or fail to reach some
conclusions because of the corresponding rule being not active in the class where
it is defined.

For example, in figure 1.1, the characterization of animal is empty because
nothing is stated about the flying abilities of animals in general. The character-
izations of the other two classes are $\langle\text{mammal}\rangle = \{\neg\text{flying}, \text{quadruped}\}$ and
$\langle\text{bat}\rangle = \{\text{flying}, \text{quadruped}\}$. So bat, although it enjoys flying, is still thought of
as being *quadruped*, by predicate inheritance, because this property is not explicitly opposed in *bat*. Also, we do not know if *bat* is *light*. The knowledge expressed in the theories would be better preserved if the rules defining *quadruped* and *light* themselves were inherited instead of just their consequences. This is the main subject of the current chapter, to which we now turn.

### 2.2 Rule inheritance

The idea behind rule inheritance is to delay the computation of conclusions until after the relevant theories for a class are somehow combined. Predicate inheritance, on the contrary, extracts the information implicit in a single theory at a time, as a function of the conclusions similarly obtainable from the respective superclasses.

Following this inspiration, the method of theory composition used by rule inheritance may start with the highest classes in the hierarchy and proceed downwards, collecting the rules in the successive lower classes. Whenever a rule head is the complementary of the one in an already collected rule, the possibility of a conflict emerges. But it may or may not become effective, depending on the bodies of the rules. To make sure that useful information is not thrown away, only when a fact is encountered are the rules for the opposite property deleted from the composition. We will denote by $\mathcal{D}_c^+$ the composition of the theory for class $c$ with the theories for its superclasses.

**Example 20** Rule inheritance for the bat example.

Recalling example 1.1, the composed theory for *animal* is just

$$\mathcal{D}_{\text{animal}}^+ = \mathcal{D}_{\text{animal}} = \{\text{light} \leftarrow \text{flying}\},$$

with no conclusions, and for *mammal* it is the union

$$\mathcal{D}_{\text{mammal}}^+ = \mathcal{D}_{\text{animal}} \cup \mathcal{D}_{\text{mammal}} = \left\{ \begin{array}{l} \text{light} \leftarrow \text{flying} \\ \text{flying} \leftarrow \neg \text{flying} \\ \text{quadruped} \leftarrow \neg \text{quadruped} \end{array} \right\},$$

with conclusions $\{\neg \text{flying}, \text{quadruped}\}$. The rule for *light* is asleep in *mammal*, but it will be awaken in *bat*, whose composed theory is the local theory $\mathcal{D}_{\text{bat}}$ plus the previously computed one for *mammal*, its sole superclass, except that the rule for $\neg \text{flying}$ is overridden by the fact $\text{flying}$, in the more specific class *bat*. The result is

$$\mathcal{D}_{\text{bat}}^+ = \left\{ \begin{array}{l} \text{light} \leftarrow \text{flying} \\ \text{flying} \leftarrow \neg \text{flying} \\ \text{quadruped} \leftarrow \neg \text{quadruped} \end{array} \right\},$$
with conclusions \{flying, tight\}. Notice that quadruped is not a conclusion for bat. The flaws pointed in predicate inheritance are overcome in this example. Inactive rules are no longer lost and dependent conclusions go away with their support. □

Unfortunately, this method is too simple-minded and does not handle the variety of situations that may occur in rule inheritance.

Example 21  The descendant method falls short.

\[
\begin{array}{c}
\text{a) } p \\
\text{b) } \neg p \leftarrow r \\
\text{c) } \neg r \\
\end{array}
\]

Figure 2.10: Exception of exception.

The composed theory for b is \(D_b^+ = \{\neg p \leftarrow r; r; p\}\). This theory is inconsistent. However, the rule \(\neg p \leftarrow r\) is clearly more specific than the fact \(p\), so it should override the latter. It is not a good idea to simply delete \(p\) from the composition. The composed theory \(D_c^+ = \{\neg r; \neg p \leftarrow r; p\}\) is no longer inconsistent because \(r\) in \(D_b^+\) is overridden by the fact \(\neg r\) in \(D_c\) and the problematic rule becomes inactive. Property \(p\) gets not opposed and is one of the conclusions for c, along with \(\neg r\).

Class b is an exception to a because it does not enjoy p, and c is an exception to b by the opposite reason. This ability to recover generic properties which were overridden by contingent conclusions, themselves unsupported in more specific places, seems to be a desirable characteristic any rule inheritance mechanism should possess.

Notice that if \(D_b = \{\neg p \leftarrow r; r\}\) was instead \(\{\neg p; r\}\), then the fact \(\neg p\) would override \(p\) in \(D_a\) and the inconsistency in \(D_b^+\) would disappear, as well as the recovering of \(p\) in class c. This shows that the two versions of \(D_b\) have really a different meaning in the HST, even if they imply the same conclusions when seen isolated.

In summary, if \(D_i^+\) contains \(p\), we are enforcing the functional character of the rules, at the cost of getting an intuitively unnecessary inconsistency in the composed theory. If \(p\) is deleted, the accent is shifted to the specificity of assertions, which loose part of its functional character, including the ability to recover from a double exception.

Of course, one could hack a solution for the inconsistency in \(D_b^+\) in the process of computing the conclusions by giving higher precedence to the local theory over the
inherited part, i.e., by working on the composed theories of the direct superclasses plus the local theory, instead of using directly the composed theory of the target class. However, the problem remains in the subclasses of \( b \) which do not contradict \( r \).

The descendant method, which works from the most generic to the most specific class, proves to be inadequate to our intuitions. What about going the other way round ascending from the most specific to more generic classes? The idea is to start with the local definition of the target class and incrementally modifying it with the theories of the superclasses, in the hope that having the more specific rules beforehand may in some way help to improve the overriding process.

An immediate disadvantage stands out: the descendant method produces the composed theories of all the classes it traverses in a single pass, while the ascendant version, being focused on a target class, gives only the respective composed theory in each pass. Worse, although the ascending method works fine with example 21, it is not in any case the solution we are seeking for, as can be easily recognized in example 22.

Example 22 A problem with the ascendant method.

![Figure 2.11: Postponed overriding.](image)

The target class is \( c \). In the first step, we set \( D^+_c,1 = D_c = \{ p \leftarrow r \} \). The second step gives \( D^+_c,2 = D^+_c,1 \cup D_b = \{ p \leftarrow r; \neg p \leftarrow r \} \) because, as we do not have \( r \) yet, we are not able to decide whether each of the rules will be necessary or not. But then \( D^+_c,3 = D^+_c,2 \cup D_a = \{ p \leftarrow r; \neg p \leftarrow r; r \} = D^+_c \) and we run again into a counterintuitive inconsistency.

This long introduction to the subject of rule inheritance was intended to convince the reader of the incapacity of the incremental methods, both ascendant and descendant, to capture the subtleties involved in the composition of hierarchically structured theories.

Example 21 suggested the interest of keeping at least two levels of preferences qualifying the rules in the composed theory. We will go a step further and retain
all the relevant hierarchic information, i.e., the computation of the conclusions will proceed directly from the family of theories of the superclasses. The relative specificity of two rules may then be always fully evaluated.

2.2.1 Specificity

We will make now a detailed presentation of our proposal for rule inheritance. To make matters precise we recall some principles.

To obtain the characterization of class \( a \) only its superclasses can be used so, in what follows, we will assume there is a target class \( a \) and restrict the hierarchy to those relevant classes. Notice that all the superstructure of \( a \) is needed and not only the immediate superclasses because a rule may produce effects anywhere in its subclasses, where the respective body becomes true, and information about its original class may be needed to settle conflicts with other rules. Predicate inheritance is more local, in the sense that it uses directly only the immediate superclasses characterizations. Trying to obtain an intermediate theory composed from the relevant theories, as a first step followed by the computation of the respective consequences, is not a good idea because the hierarchic relation gets masked in the process. We will need the original family of theories of the target superclasses \( D = (D_c)_{a \leq c} \) plus the hierarchic relation. The problem here is how to define a notion of specificity to guide the overriding process.

**Definition 46 (Specificity)** Class \( b \) is more specific than \( c \) iff \( b < c \), i.e., there is a path from \( b \) to \( c \).

The notion of specificity we use is simply the transitive closure of the hierarchic relation. Notice that the specificity relation is a partial order and so some pairs of classes may be unrelated (like republican and quaker in figure 2.12), creating the conditions for neutralization (of pacifist, relatively to nixon).

![Diagram](image)

\(<\text{nixon}> = \emptyset\)

Figure 2.12: Nixon dilemma.

A conclusion is the head of a rule whose body has already been concluded and found to be true.
Definition 47 (Neutralization) *If two opposite conclusions originate in two incomparable classes, they neutralize each other and none is true.*

If a contradiction evolves between properties coming from disjoint lines of inheritance, no matter how long these lines are, then specificity is of no help. There is a general understanding that none should prevail and neutralization (mutual annihilation) of the pair is recommended. Such a HST is ambiguous. As `republican ≠ quaker` and `quaker ≠ republican`, none is more specific than the other and then neither `pacifist` nor `¬pacifist` is true for `nixon`. They are both undefined implying a 3-valued or partial semantics for atomic properties in characterizations.

When the conflicting rules belong to the same class, neutralization still applies. No class can be more specific than itself. In this limit case, we decided to maintain the definition for three reasons:

- uniformity — otherwise a special case for this particular kind of ambiguity would be needed;
- generality — we could rule out HSTs leading to contradictions in a single class, dubbing them inconsistent, but then the remaining information they contain would be lost; an ambiguity generated in two unrelated classes is not essentially different from an ambiguity locally unveiled, but possibly supported by previous conclusions obtained in diverse places;
- expressivity — stating two contradictory facts in a class can be understood as forcing the corresponding property to be undefined, independently of whatever is inherited.

Definition 48 (Overriding) *If a conclusion originates in a class b and its opposite in a class c, such that b is more specific than c, then the former conclusion overrides the latter.*

For an example of this fundamental mechanism of exceptions, see figure 2.13 (adapted from [San86]). With respect to `royal-elephant`, `¬gray` overrides `gray`, which is stated in a superclass.

Notice also that there may be several paths between two classes. Assume that `p` overrides `p` along one path. The non occurrence of `p` in an alternative path does not prevent the overriding. See `¬gray` in `royal-elephant` overriding `gray` in `elephant`, with respect to target `clyde`, despite the path through `african-elephant`. As `african-elephant` is `gray`, it could seem that this is a neutralization situation. In predicate inheritance that would be the case, as well as in the ascendant and descendant versions tried before. Due to the incremental and local nature of those
mechanisms, *clyde*’s color would remain undecided. The proposal we are now presenting for rule inheritance has a more global character and so it is meaningful to use, as a disambiguating criterion, the relative position of possibly distant classes to be able to detect common superclasses in different paths.

The two principles just presented are meant to keep the characterizations consistent by resolving the conflicts in the HSTs, due to exceptions and to ambiguities, based on relative specificity.

Specificity is determined by the topology of the hierarchy and the actual number of classes along a given path is immaterial for this criterion. This contrasts with approaches where specificity is measured by the distance in terms of number of classes but agrees with inferential distance, as extended with off-path preemption [San86].

Path-based methods build sets of permitted paths (from which conclusions are then extracted) and to deal with conflicts they must resort to the notions of conflictors and preemptors. In our approach, the inheritance paths are already available and basic conflicts among properties assume the simpler forms of neutralization and overriding. Of course, complex interactions with multiple neutralizations and overrides may arise.

Inheriting rules carries along a further difficulty. The specificity criterion, applied to the rules for a given property, is able to select the more specific. But the corresponding bodies may have properties for which there are rules with the inverse ordering. Classifying one of the overall arguments as more specific than the other seems not to be intuitively founded. It is even possible that the more specific rule never becomes active, rendering any overriding only apparent.

**Example 23** *Jumping among levels of specificity.*

At first sight it seems that, in figure 2.14, property $p$ can be inferred from rules $p \leftarrow r$ and $r$, therefore overriding $\neg p$. But, following another line of reasoning, from $t$, $s \leftarrow t$, and $\neg r \leftarrow s$ it is possible to obtain $\neg r$, overriding $r$. This way $p$ is not derived, leaving room for $\neg p$. 

\[\square\]
2.2 Rule inheritance

Figure 2.14: A convoluted example.

So, the following condition is added to the notion of specificity to soften over-
riding: only facts have the power to override. Rules are not considered reliable
enough, at least until the respective bodies are known to be true, thus equating
them to (conditional) facts. This principle points to a bottom-up approach to the
computation of the characterizations, closer in spirit to the argument construction
paradigm [HTT90] than to the property flow metaphor that inspired predicate inher-
heritance. Specificity is used in a dynamic way, selecting lines of reasoning, instead
of statically assigning levels of preference to rules.

We see the rules in theories as definitions for the properties in the respective
heads, expressing dependencies between properties. A conclusion which is not
contradicted must be accepted. Overriding is not arbitrary. It is not necessary to
search inside the bodies in order to check that a property cannot be overridden. It
is enough to note that there are no rules for its opposite. This view is consistent
with seeing the arrow as an inference rule, for which the contrapositive does not
hold. In example 23, we can safely conclude from \( t \) and \( s \leftarrow t \) that \( s \) holds for \( a \)
because there is no rule for \( \neg s \). However, in classical logic, using the contrapositive,
from \( r \) and \( \neg r \leftarrow s \), it could be inferred \( \neg s \).

**Example 24 Inconsistency or ambiguity?**

Figure 2.15: A local neutralization.

In figure 2.15, given that \( r \) and \( s \) hold, the knowledge expressed is ambiguous
about \( p \). To avoid inconsistency, \( p \) is neutralized. It may seem against the spirit
of specificity that the rules in the lower class be the ones which are ignored. But
according to the notion of specificity developed in this section, the comparison is
between lines of reasoning. Those at stake here are followed without problems (r and s are unrelated) until the contradiction on p is found, but then specificity is of no help. Anyway, the local theory in a has no consequences by itself, so inheriting r and s from b is not contradicting anything.

The alternative would be to go through the contrapositives and override r, or s, or both. This procedure is, in general, nondeterministic and decouples overriding from the simple notion of contradiction.

The characterization does not commit to any unsupported conclusions and it is not blocked by the occurrence of conflicts, neither in a multiple inheritance situation nor locally generated. The elimination of inconsistencies permits the access to the remaining information and avoids the above mentioned phenomenon of explosion in the characterization, where every formula is derivable from a contradiction.

In summary, the rationale behind our proposal is based on the following principles:

- The only classes with theories relevant to the computation are the target class and its superclasses.
- A conclusion which is not contradicted must be accepted.
- The contrapositive does not hold.
- If a conclusion is more specific than an opposite conclusion it overrides the latter.
- If, given two opposite conclusions, none is more specific than the other they are both neutralized.
- Facts alone are definitive about overriding.

Rule inheritance, as being proposed here, is not driven by the hierarchy, but by the dependencies among properties expressed in the theories. It is neither ascendant nor descendant, but rather deals globally with the relevant classes. It is bottom-up in the sense that it starts from the facts, wherever they are, and proceeds until the more elaborated consequences of the rules are reached, much as if there was a single big theory, but always taking into consideration the specificity relation. The hierarchic relation is in no way affected by properties and acts as a static backbone for their nonmonotonic inheritance. To insist on ascribing only to facts the ability to override lends a dynamic incidence to the notion of specificity.
2.2.2 Fixed-point definition of the characterization

A proposal follows for the rule inheritance mechanism [DP93] based on the principles just presented. The goal is to define, in the context of a given HST \( \langle C, P^+, \prec, D \rangle \), the set of conclusions \(<a> \subseteq P\), taking into account the dependencies between properties expressed by the rules, for any target class \( a \in C \).

The HST will be implicit in the following definitions. A class can inherit only from its superclasses. The rest of the system can in no way affect the properties holding for it. We call relevant theories for a target \( a \) the family of class definitions \( D = (D_c)_{a \leq c} \). The elements of this family are partially ordered by the reflexive transitive closure \( \leq \) of the hierarchic relation \( \prec \) and have a least element, the target class itself.

The characterization \(<a>\) of a target class \( a \) is defined through an iterated fixed-point [PP90]. The outer fixed-point contains the characterization. The inner fixed-point, for each step of the outer fixed-point, adds the indisputable conclusions, i.e., those neither overridden nor neutralized, of the theory simplified with respect to the previous set of conclusions.

**Definition 49 (Characterization)** The following set of equations is a bottom-up definition of the characterization of a target class \( a \).

\[
\begin{align*}
T_D(I) & = \{p \mid \exists c : p \leftarrow Q \in D_c, Q \subseteq I, \neg\exists b \prec c : \bar{p} \in D_b\} \\
T^*(D) & = T_D \uparrow w \\
(D[I]_c & = \{p \leftarrow Q \in D_c | \bar{p} \not\in I\} \\
\Delta_D & = T^*(D[T^*(D)]) \\
(D/J)_c & = \{p \leftarrow (Q \setminus J) | p, \bar{p} \not\in J, p \leftarrow Q \in D_c, \bar{Q} \cap J = \emptyset\} \\
S_D(J) & = J \cup \Delta(D/J) \\
<a> & = S(D_c)_{a \leq c} \uparrow w
\end{align*}
\]  

(2.11)  
(2.12)  
(2.13)  
(2.14)  
(2.15)  
(2.16)  
(2.17)

The basic idea is to apply a modified immediate consequence operator \( T_D : \varphi(P) \rightarrow \varphi(P) \) to the relevant theories \( (D) \). The modification consists in blocking a conclusion of a rule defined in a class \( c \) if there is a fact below \( c \) stating the opposite. In the definition, \( I \) is an interpretation, i.e., a set of positive and negative properties. The first part of the condition corresponds to the usual immediate consequence operator, working on the union of all the definitions for the superclasses. However, the rules must not loose their origin because the second part expresses the principle that facts override upper definitions, thus inhibiting some of the conclusions.

Using again the renaming trick on the negative properties, \( D \) corresponds to a positive logic program. As only the facts are able to override, their action is
independent of the set $I$. The operator $T_D$ is monotonic and has a least fixed-point obtainable in $w$ steps and denoted by $T^*(D)$. It associates to each relevant theory $D$ the set of all its consequences, except those overridden by facts. It may contain contradictions and possibly some properties which are not directly challenged but nevertheless depend on contradictions. Nothing else can be concluded. This puts an upper bound on the set $<a>$.

To get a lower bound, the theory $D$ is restricted to the rules whose heads are not involved in the conflicts patent in the upper bound. The restriction operator $D[I]$ performs an implicit neutralization on the theories, with respect to a set of properties $I$. The restricted program $D[T^*(D)]$ is void of conflicting rules. All the properties derived from it can’t be opposed and must be kept. But it is not enough to remove all the generated contradictions. A second application of $T^*$ is necessary to wipe out the consequences which become unsupported after such removal. The set $\Delta_D$ contains only the indisputable conclusions that can be obtained from the current theory and imposes a lower bound on $<a>$.

The intended set lies between the two bounds $T^*(D)$ and $\Delta_D$.

Obtaining $\Delta_D$ completes one step of the outer fixed-point construction. The next step is similar but starts with a simpler theory where the knowledge of $\Delta_D$ is taken into account.

The quotient $[GL88]$ of a program $D$ by a set of properties $J$ is the residual part of the theory which is able to generate new conclusions beyond $J$. Rules for properties already settled become irrelevant, rules which can’t be activated are dropped, and properties already acquired are deleted from the remaining rule bodies. The crucial point here is that some rules may become facts after the division and then be enabled to do some overriding.

The whole procedure, denoted by the safe conclusions operator $S_D$, may be iterated. Some of the previous conclusions are regenerated, under the new conditions, and a new increment $\Delta_D$ of granted properties is obtained. This narrows the distance between the extreme sets. As $\Delta_{(D/J)}$ can only contain new literals not in $J$ and $J$ can only grow, it eventually becomes empty and the operator $S_D$ reaches a fixed-point, which is the characterization of the target class $a$, $<a>=S_D(<a>)$. The existence of such a least fixed-point is guaranteed by the continuity of the operator.

**Theorem 50 (Continuity)** The safe conclusions operator $S_D$ is continuous.

**Proof:**

Consider an infinite sequence

$$J_0 \subseteq J_1 \subseteq \ldots$$
of interpretations and suppose that

\[ p \in S_D(\bigcup_{n=0}^{\infty} J_n) = (\bigcup_{n=0}^{\infty} J_n) \cup \Delta(D/\bigcup_{n=0}^{\infty} J_n). \]

Due to the definition of quotient (2.15), if \( r \in J \) then \( r \notin \Delta_D/J \). So,

\[ \bigcup_{n=0}^{\infty} J_n \text{ and } \Delta(D/\bigcup_{n=0}^{\infty} J_n) \]

are disjoint sets.

If \( p \in \bigcup_{n=0}^{\infty} J_n \) there is an \( n \) such that \( p \in J_n \) and, by (2.16), \( p \in S_D(J_n) \).

Otherwise, \( p \in \Delta(D/\bigcup_{n=0}^{\infty} J_n) \). So, by (2.14), there is a rule \( p \leftarrow Q \in (D)_c \) such that

\[ Q \subseteq \bigcup_{n=0}^{\infty} J_n \cup T^*(D/\bigcup_{n=0}^{\infty} J_n) \]

\[ \overline{p, Q} \cap T^*(D/\bigcup_{n=0}^{\infty} J_n) = \emptyset. \]

As the body \( Q \) and the theories \( D \) are finite, there is an order \( n \) after which

\[ Q \subseteq J_n \cup T^*(D/J_n) \text{ and } \overline{p, Q} \cap T^*(D/J_n) = \emptyset. \]

Then \( p \in \Delta(D/J_n) \) and therefore \( p \in S_D(J_n) \).

So \( p \in \bigcup_{n=0}^{\infty} S_D(J_n) \) and the continuity of \( S_D \) is proved.

The set of equations (2.11-2.17) is a bottom-up statement of the semantics attached to a class in the HST. The iterated fixed-point contains the set of properties one should expect to hold for it. Such a definition is good for precisely establishing the intended meaning of a HST, but is not practical from a computational viewpoint, which requires a goal-directed procedure. This subject is left to the next chapter.

**Example 25** The bottom-up procedure.

In figure 2.16, the relevant theories for the target *cetaceous* are

\[ D = \{D_{cetaceous}, D_{mammal}, D_{aquatic}, D_{animal}\} = D^0. \]

The first application of the immediate consequences operator gives only the facts

\[ I^{01} = T_{D^0}(\emptyset) = \{milk, \neg milk, fin, \neg swim\}. \]

The second application produces already a fixed-point

\[ I^{02} = T_{D^0}(I^{01}) = \{milk, \neg milk, fin, swim, \neg swim\} = T^*(D^0). \]
The upper bound $T^*(D^0)$ contains every property which may conceivably be generated for cetaceous, including two contradictions. These are temporarily neutralized in the restricted theory $D^0[T^*(D^0)]$, whose sole non-empty definition is $(D^0[T^*(D^0)])_{aquatic} = \{\text{fin}\}$. The first lower bound is the corresponding set of conclusions (in the beginning nothing is known, so $J^0 = \emptyset$)

$$\Delta_{D^0} = T^*(D^0[T^*(D^0)]) = \{\text{fin}\} = S_D(\emptyset) = J^1.$$

This is the first inner fixed-point, which is used to simplify the current theory through an operation called division. The quotient $D^1 = D^0/J^1$ differs from $D^0$, in that the fact $\text{fin}$ disappears, since it is now known to be true, and the rule $\text{swim} \leftarrow \text{fin}$ becomes the fact $\text{swim}$ and acquires the strength to override $\neg \text{swim}$ in animal. So, the immediate consequence operator will produce

$$I^{11} = T_{D^1}(\emptyset) = \{\text{milk}, \neg \text{milk}, \text{swim}\} = T^*(D^1).$$

Now, there is only one contradiction which is neutralized by the restriction operator, leaving the fact $\text{swim}$ alone in aquatic.

$$\Delta_{D^1} = T^*(D^1[T^*(D^1)]) = \{\text{swim}\}$$

The second iteration of the safe conclusions operator is

$$S^2_D(\emptyset) = S_D(S_D(\emptyset)) = S_D(J^1) = J^1 \cup \Delta_{(D/J^1)} = J^1 \cup \Delta_{D^1} = \{\text{fin}\} \cup \{\text{swim}\} = J^2.$$ 

Now, the quotient $D^2 = D/J^2 = D^1/J^2$ (generally, $D^n = D/J^n = D^{n-1}/J^n$) retains just the contradictory pair milk $-$ milk, which is neutralized, originating an empty theory and an empty set $\Delta_{D^2}$ of conclusions. Nothing is added to $S^2_D(\emptyset)$ and so it is the sought fixed-point.

$$<\text{cetaceous}> = S^2_D(\emptyset) = \{\text{fin}, \text{swim}\}.$$

Notice that, of the two contradictions found in the first upper bound, one was resolved by overriding and the other by neutralization. The upper and lower bounds got closer along the iteration, but did not converge completely. The difference is precisely the neutralized pair.
Example 26 Jumping among levels again.

This example shows how rule inheritance behaves with respect to the example 23, where the lines of reasoning are tangled together. For target class $a$ the relevant theories are $D = \{D_a, D_b, D_c\}$. The two bounds on $\langle a \rangle$, starting with $J^0 = \emptyset$ and $D^0 = D$, are

\[
T^*(D^0) = \{t, \neg p, r, p, u, s, \neg r\} \\
\Delta_{D^0} = T^*(D^0[T^*(D^0)]) = \{t, s\}.
\]

Notice that $\neg t$ is not generated by $T^*(D^0)$ because it is blocked by $t$, and that $u \not\in \Delta_{D^0}$, although there are no rules for $\neg u$, because it depends on $r$, a conflicting property. Taking $J^1 = J^0 \cup \Delta_{D^0}$ for granted, the program can be simplified (see figure 2.17):

\[
D^1 = D/J^1 \\
T^*(D^1) = \{\neg p, \neg r\} = \Delta_{D^1}.
\]

The transformation of $\neg r \leftarrow s$ in class $b$ into a fact $\neg r$ forces the overriding of $r$ in $c$, leaving the rule $p \leftarrow r$ inactive in class $a$. As no contradictions were generated in $T^*(D^1)$ the increment $\Delta_{D^1}$ coincides with it, subsequently reaching the fixed-point $\langle a \rangle = J^2 = J^1 \cup \Delta_{D^1} = \{t, s, \neg p, \neg r\}$.

Repeating the same procedure for the other classes, $\langle b \rangle = \{\neg t, u, \neg p, r\}$ and $\langle c \rangle = \{u, r\}$. In this example, there are more exceptions than defaults (compare $\langle a \rangle$ and $\langle b \rangle$). It is not strange that the decoupling phenomenon [Tou86] occurs here. For instance, $s$ holds for $a$ and not for $b$, but can only reach class $a$ through class $b$. The problem is that class $b$ does not have the materials to make $s$ actual, while $a$'s are a special case of $b$'s which have that capability. □
The compilation of the conclusions for c's superclasses is not useful to obtain \(<c>\), if dependencies among properties expressed in the rules are to be captured. The method works as if the rules were pulled to the target class and then used, instead of producing effects in their own original classes. Fetching the rules may be blocked if a contradictory fact is found in the set of paths between the rule class and the target class.

The version of rule inheritance presented refers directly to the family of theories in superclasses, instead of composing, in an intermediate step, an equivalent single theory, from the perspective of the target class. Still, the concept of composed theory is meaningful, but now as a result of the inheritance mechanism.

**Definition 51 (Composed theory)** The composed theory \(D^+_a\) of a class a is the union of the relevant theories, after deleting the rules whose heads are not in \(<a>\).

The rule inheritance mechanism is somewhat complex and overwhelmed with details, but can be interpreted in a more abstract way as the rules in \(D^+_a\) collectively overriding the other rules in \((D_e)_{a \leq e}\). It is not abusive to reduce the whole process to an overriding equivalent because even neutralization, the other basic operation involved, has a reading as a mutual overriding without winner.

The characterization \(<a>\) is the least model of the composed theory \(D^+_a\), although this result is not of much significance because the satisfaction relation, underlying the notion of model, is restricted here to the rules which are not overridden.

**Lemma 52 (Supportedness)** For every property p in the characterization of a class a there is at least one rule \(p \rightarrow Q\) in its composed theory \(D^+_a\) such that the body Q is contained in that characterization.

**Proof:**

The characterization \(<a>\) is the least fixed-point of the operator \(S_D(J)\). In each step, the set \(J\) is augmented by the least fixed-point \(\Delta_{D/J}\) of the operator \(T_{D/J}\) applied to a subset of the rules in \(D/J\). For every property in \(\Delta_{D/J}\) there is a rule in \(D/J\) such that the properties in its body are also in \(\Delta_{D/J}\). This recursion ends with the facts as they have empty bodies. A least fixed-point does not contain unfounded conclusions, i.e., conclusions depending on themselves, so the recursion is finite.

To relate \(D/J\) with the initial HST \(D\) we must note that a fact in \(D/J\) may correspond to a rule with a non-empty body in \(D\). However, the definition of the quotient operation implies that, in such a situation, the deleted properties from the body already belong to \(J\). The original rule in \(D\) also belongs to \(D^+_a\) because
the composed theory contains all the rules in the relevant classes \( D \), for each of the properties in \(<\alpha>\).

\[ \text{\textbullet} \]

**Theorem 53 (Consistency)** Characterizations are consistent.

**Proof:**

The characterization \(<\alpha>\) is the union of a sequence of least fixed-points \( \Delta_{D/J} \) of the immediate consequence operator applied to \( D/J[T^*(D/J)] \). If a contradictory pair \( p - \overline{p} \) is generated by \( T^*(D/J) \), the corresponding restriction is free of rules both for \( p \) and for \( \overline{p} \). So, none of these subsets of the rules in \( D/J \) is able to generate contradictory pairs.

If a property \( p \) is generated by \( T^*(D/J) \) and \( \overline{p} \) is not, then \( p \) is included in \( \Delta_{D/J} \) and all rules for \( p \) and for \( \overline{p} \) are absent from the subsequent \( D/J \), due to the definition of quotient. Each property in \(<\alpha>\) belongs to exactly one set \( \Delta_{D/J} \). So, contradictory properties can not come from different sets \( \Delta_{D/J} \).

The sets of properties \( J^n \cup T^*(D/J^n) \), generated along the bottom-up iteration, form a monotonic non-increasing sequence. The set \( J \) monotonically increases along the iteration of \( S_D \). The HST \( D/J \) can only become smaller, either by the deletion of rules that became useless or by the deletion of properties in the remaining rules which were incorporated in \( J \). If they were incorporated in \( J \) then they were the heads of rules with bodies also in \( J \). So every property that is generated from one of the HST \( D/J \) must also be produced by the generating part of \( T_D \), applied to the initial \( D \). The inhibiting part of \( T_D \) does not decrease faster than the generating part, i.e., if a fact \( p \) inhibits a conclusion \( \overline{p} \) then when \( p \) is deleted by the division, so is \( \overline{p} \). Thus every property in \( \Delta_{D/J} \) must be among those in \( T_D \).

**Lemma 54 (Nesting)** Let \( I \) and \( J \) be consistent sets of properties such that \( I \subseteq J \) and \( X = J \setminus I \subseteq T^*(D/I) \). Then

\[
J \cup T^*(D/J) \subseteq I \cup T^*(D/I).
\]

**Proof:**

If \( p \in J \) then \( p \in I \cup X \subseteq I \cup T^*(D/I) \). It is more interesting the case \( p \notin J \).

According to (2.15), the theories \( D/J \) are contained in \( D/I \) in the following sense \( (Q' \cap X = \emptyset, Q'' \subseteq X) \):

\[
p \leftarrow Q' \in (D/J)_b \quad \Rightarrow \quad p \leftarrow Q', Q'' \in (D/I)_b \quad (2.18)
\]

\[
p \leftarrow Q', Q'' \in (D/I)_b \quad \Rightarrow \quad p \leftarrow Q' \in (D/J)_b \vee (Q' \cup \{p, \overline{p}\}) \cap X \neq \emptyset. \quad (2.19)
\]
We prove that $T^*(D/J) \subseteq T^*(D/I)$ by induction on the number of applications of $T_{D/J}$.

Base case: $T_{D/J}^0(\emptyset) = \emptyset \subseteq T^*(D/I)$.

Inductive step: if $p \in T_{D/J}^k(\emptyset)$ then there is a rule $p \leftarrow Q' \in (D/J)_b$ such that $Q' \subseteq T_{D/J}^{k-1}(\emptyset)$, by the supportedness of (2.11). Besides, there is no fact $\overline{p} \leftarrow \in (D/J)_c, c < b$. Using (2.19), this implies that there is also no fact $\overline{p} \leftarrow \in (D/I)_c$ because $\overline{Q'} = \emptyset$ and $\overline{p}, \overline{\overline{p}} \not\in X$, otherwise, by (2.15), no rule for $p$ would exist in $D/J$. By (2.18), there is a rule $p \leftarrow Q', Q'' \in (D/I)_b$ and $Q'' \subseteq X \subseteq T^*(D/I)$. As, by hypothesis, $Q' \subseteq T^*(D/I)$ then $p \in T^*(D/I)$.

Corollary 55 (Upper bound) Every property that can be concluded from a HST $D$ is contained in the first application of the consequences operator,

$$<a> \cup T^*(D/ <a>) \subseteq T^*(D).$$

Proof:

In the step $n + 1$ of the bottom-up iteration (2.16) the set $J^n$ is increased by the properties in $\Delta_{D/J^n}$, which, according to (2.13), are a subset of $T^*(D/J^n)$. So, lemma 54 may be used by setting $I = J^n$ and $X = \Delta_{D/J^n}$ what implies, by (2.16),

$$J = I \cup X = J^n \cup \Delta_{D/J^n} = J^{n+1},$$

for $n = 0, 1, \ldots$:

$$J^{n+1} \cup T^*(D/J^{n+1}) \subseteq J^n \cup T^*(D/J^n). \tag{2.20}$$

From (2.16), (2.17), and the finitude of $<a>$, there is an order $n'$ at which $<a> = J^{n'} \supseteq J^n$. By the transitivity of (2.20),

$$<a> \cup T^*(D/ <a>) \subseteq J^n \cup T^*(D/J^n) \subseteq J^0 \cup T^*(D/J^0) = T^*(D)$$

proves the intended result.

Theorem 56 (Restricted HST) The characterization is a subset of the consequences of the HST restricted by the characterization itself,

$$<a> \subseteq T^*(D/ <a>).$$

Proof:

We already know, from corollary 55, that $<a> \subseteq T^*(D)$. Notice that $D/ <a>$ is $D$ without the rules whose heads contradict $<a>$. Those rules do not interfere with the computation of $<a>$ (remember, no contrapositive effects). Their heads are not needed directly as they can not belong to $<a>$ due to its consistency (theorem 53). They are not indirectly required too, to support properties in $<a>$ because, by lemma 52, all such properties are in the characterization.
2.2 Rule inheritance

Theorem 57 (Global overriding) The characterization $<a>$ is the least model of the composed theory $D^+_a$.

Proof:

The composed theory $D^+_a$ is free of the rules that contradict $<a>$ (like $D[<a>$) and also of those that, despite they do not clash with $<a>$, are nevertheless irrelevant to it. The purged rules do not contribute to the characterization neither directly nor indirectly, by an argument that follows the proof of theorem 56. As there are no contradictions in the rules left, there are no overridings too. So the hierarchic structure is no longer needed and the immediate consequence operator $T_{D^+_a}$ reduces to the usual $T_P$ operator of Logic Programming. This means that $<a>\subseteq T_{D^+_a} \uparrow \omega$. On the other side, as the heads of the rules left all belong to $<a>$, we must also have $T_{D^+_a} \uparrow \omega \subseteq <a>$.

So, using the standard result of the equivalence between the least fixed-point of $T_P$ and the least model semantics of a definite logic program, we may conclude that $<a>$ is the least model of $D^+_a$ in the Fitting sense.

Notice that it is essential to delete the rules that correspond to neutralizations or depend on them in order to allow the application of the standard immediate consequence operator and keep the consistency. The rules that are inactive in the HST will remain so. Thus their elimination is not of much importance.

2.2.3 Behavior of the HST

The results in this subsection highlight some features of HST which help to make more precise its shape as modeled by rule inheritance.

2.2.3.1 Restricted monotonicity

The semantics presented for HST is essentially nonmonotonic, i.e., adding a new rule to a theory may force a previous conclusion to be dismissed, if it creates an overriding or neutralization situation either directly or involving one of the rules it may activate. However, this nonmonotonicity is not completely unwieldy. Some features of HST soften its behavior. In particular, reifying derived conclusions as actual assertions may be done in the following situations.

Theorem 58 (Hierarchic invariance) If a class is more specific than another, $a < b$, then adding the direct link $a$-$b$ to the hierarchic relation does not affect the semantics.
Proof:

The semantics of definition 49 does not depend directly on the hierarchic relation $\prec$ but only on the specificity partial ordering $\prec$. If $a \prec b$ then there is a path $a \prec \ldots \prec b$ and the addition of the alternative path $a \prec b$ just enforces that observation.

In other words, transitive, redundant links in the hierarchic relation are innocuous. Achieving this result explains a posteriori why we chose, to guide inheritance, the more abstract notion of specificity instead of a path-based criterion.

Notice that inheritance nets do not possess this feature, called by [Tou86] generic stability because for them all the links are defeasible. Inheritance nets enjoy just atomic stability, i.e., if a conclusion is found to hold for a leaf node (an individual), then the corresponding link may be safely added to the net. The defeasible part of HST has similar problems.

**Theorem 59 (Property invariance)** If $c$ is a minimal class under specificity, adding (part of) the characterization $\prec c$ to the class definition $\mathcal{D}_c$ does not change the semantics of the HST.

Proof:

If $p$ is already a conclusion for target $c$, any rule for $\overline{p}$ is eventually overridden. Adding the fact $p$ directly to the class $c$ can only speed up the process. As $c$ is required to be minimal, there is no need to consider the effect on other classes.

This theorem does not hold in general for classes which are not leaves in the hierarchy because adding facts would harm the dependency information and resurrect some of the behaviors of predicate inheritance with respect to the classes below the one changed. Permanently asserting in a class a conclusion, whose support may be overridden in a lower class, changes the semantics. Intermediate conclusions, during a computation, may be used as lemmas but only if they are just temporarily asserted and circumscribed to the context of the target class.

In the case of predicate inheritance redundancy in the hierarchic relation was forbidden because, as it does not use preemption, alternative paths may cheat specificity. Testing the origin of rules is not in the spirit of predicate inheritance that receives preprocessed results and not the original rules.

The present discussion reminds the concept of cumulativity [Mak88]. The HST has the ability to cumulate a conclusion into the premises (within the above mentioned limitations) without changing the other conclusions.

$$A \subseteq B \subseteq C(A) \Rightarrow C(A) = C(B)$$  \hspace{1cm} (2.21)
2.2 Rule inheritance

$C(A)$ is an inference operation, a kind of special nonmonotonic consequence operation. In the case of the HST, it may contain the characterization of a class and the corresponding set of rules which are not overridden. Any formula in $B \setminus A$ is also a consequence of $A$. Cumulativity says that including them in $A$, which gives $B$, does not change the power of the premises, as $C(A)$ remains equal to $C(B)$.

However, due to the proper nature of overriding, HST does not respect inclusion.

$$A \subseteq C(A)$$ \hspace{1cm} (2.22)

If some of the rules in $A$ are overridden, they will not show up in $C(A)$. The condition (2.22) is usually assumed in the studies on cumulativity [Mak88, FL93], turning many of their results inapplicable to HST.

2.2.3.2 Dealing with ambiguity

We have already related the principle of neutralization to the ontological problem of ambiguity\(^2\). We think that a reasoning method should not commit itself to one of the possibilities when a genuine ambiguity arises. A theory is ambiguous on a property $p$, with respect to a given semantics, if it has support for both $p$ and $\overline{p}$ and no criteria to choose one of them. A possible interpretation for this situation is that some of the typical individuals may enjoy $p$ and others $\overline{p}$ thus preventing a unique assignment for the whole typical subset. By this reason we adopt the skeptical strategy of neutralization. The indisputable properties are collected in the characterization $<a>$. To be able to name all the properties, including those involved in neutralizations, we introduce the following definition.

**Definition 60 (Weak characterization)** Let $D$ be the family of relevant theories with respect to target class $a$. The class definitions in the family $\overline{D} = D / <a>$ are called the residual theories. The set of properties $T^*(\overline{D})$ is called the ambiguous residue of class $a$.

The union of the characterization of a class $a$ with the respective ambiguous residue is called the weak characterization of $a$, $\langle a \rangle = \langle a \rangle \cup T^*(\overline{D})$.

As $\langle a \rangle$ is a fixed-point of $S_D$ nothing else can be safely concluded from the program $D$. Overriding depends on the safe conclusions and so nothing can be overridden beyond the characterization. The residual theories $\overline{D}$ retains only

\(^2\)Some other results on the interaction of overriding and neutralization are left to the comparison between HST and related work.
those rules which support ambiguities, depend on them or are unfounded. So
$T^*(\overline{D})$ will contain those properties for which there is a supporting evidence but
nevertheless depend on ambiguity. The weak characterization $\alpha$ is the union of
these with $a$. Notice that, from corollary 55, we already know that the weak
characterization is contained in the first application of the consequences operator,
$\alpha \subseteq T^*(D)$.

Lemma 61 (Simplification) If a property belongs to the characterization then
there is an order in the bottom-up iteration after which it is no longer generated,

\[ \forall p \in \alpha \exists k : \forall n \geq k, p \notin T^*(D/J^n). \]

Proof:

If $p \in \alpha$ then, by (2.16-2.17), $p \in \bigcup_{n=0}^{\infty} J^n$ because $\Delta_{\overline{D}} = \emptyset$. As $J^n$ is an
increasing sequence starting with the empty set, there is a least order $k$ at which
$p$ starts to belong to the sequence. The division operation $D/J$ removes from the
theories in $D$ all the rules whose heads or their complements are in $J$. So for all
$n \geq k$ the theories $D/J^n$ do not contain rules able to generate a $p \in J^k$.

Corollary 62 (Disjointness) The characterization and the ambiguous residue
are disjoint,

\[ \alpha \cap T^*(\overline{D}) = \emptyset. \]

Proof:

This is immediate from lemma 61 setting $k'$ equal or greater than the greatest
$k$ for the properties in $\alpha$.

Lemma 63 (Blocking) $p \in \alpha \Rightarrow \overline{p} \notin \alpha$.

Proof:

Using the consistency of the characterization (theorem 53) $p \in \alpha \Rightarrow \overline{p} \notin \alpha$.
Due to the construction of $\alpha$, the only way to get $p \in \alpha$ with $p \in \alpha$ would
be through $T^*(\overline{D})$ but, by (2.15), the quotient $\overline{D} = D/\alpha$ does not contain rules
neither for $p$ nor for $\overline{p}$ whenever $p$ is in the divisor $\alpha$.

Corollary 64 (Filtering) The characterization is contained in the filtered weak
correlation,

\[ \alpha \subseteq \Theta \alpha. \]
2.2 Rule inheritance

Proof:

By construction (see definition 60), \(<a> \subseteq \overline{\sim a}\). By lemma 63, no complementary pair in the weak characterization involves properties in the characterization. The filtering operation removes the complementary pairs from its argument. So, it only affects properties of \(\overline{\sim a}\) that are not in \(<a>\).

The previous statement is not an equality because there may be properties in \(\overline{\sim a}\) that are not contradicted but nevertheless depend on a contradictory pair.

Lemma 65 (Ambiguities) The consequences obtained from the restriction of the relevant HST by a characterization coincide with the weak characterization, i.e., \(\overline{\sim a} = T^*(D[<a>]\).

Proof:

The implications (2.23) follow from the definitions of restriction (2.13) and quotient (2.15). Notice that properties in the bodies \(Q, Q'\) are partitioned such that \(Q \cap <a> = \emptyset\).

\[
p \leftarrow Q \in (\overline{D})_b \Rightarrow p \leftarrow Q, Q' \in (D[<a>])_b \land Q' \subseteq <a>
\]
\[
p \leftarrow Q \in (\overline{D})_b \Leftarrow p \leftarrow Q, Q' \in (D[<a>])_b \land Q' \subseteq <a> \land p, \overline{Q} \cap <a> = \emptyset
\]

(2.23)

Every rule in \(\overline{D}\) corresponds to a rule in \(D[<a>]\). It is already known, from theorem 56, that \(<a> \subseteq T^*(D[<a>]\). The excess rules in \(D[<a>]\), which do not have a correspondent in \(\overline{D}\), are irrelevant because either they are inactive by having a body that contradicts \(<a>\) or they have a head already in \(<a>\).

The close relationship between both HSTs allows the proof of the following stronger statement, where \(T^*_D(I) = I\),

\[<a> \cup T^k_D(\emptyset) = T^k_D(\emptyset) \quad k = 0, 1, 2, \ldots\]

(2.24)

Due to the lemma 52, to the inclusion \(\mathcal{D}^+_a \subseteq \cup_b (D)\) and that among the excess rules in \(D\) there are no facts for \(\overline{p}\), such that \(p \in <a>\), it is concluded

\[<a> \subseteq T^k_D(\emptyset) \quad k = 0, 1, 2, \ldots\]

(2.25)

Equation (2.24) is proved by induction on the number \(k\) of applications of the immediate consequences operator. The base case is for \(k = 0\),

\[<a> \cup T^0_D(\emptyset) = <a> \cup \emptyset = <a> = T^0_D(\emptyset)\]
The induction step.

\[ p \in T_{\overline{D}}^{k+1}(\emptyset) \iff \]

by \((2.11)\)

\[ \exists p \leftarrow Q \in (\overline{D})_b, Q \subseteq T_{\overline{D}}^k(\emptyset), \neg \exists p \leftarrow \emptyset \in (\overline{D})_c, c < b \iff \]

by the hypothesis, \((2.23)\) and \((2.25)\)

\[ \exists p \leftarrow Q, Q' \in (D[<a>])_b, Q \subseteq T_{\overline{D}[<a>]}(\emptyset), Q' \subseteq <a> \subseteq T_{\overline{D}[<a>]}(\emptyset), (p, \overline{Q}) \cap <a> = \emptyset, \]

\[ \neg \exists p \leftarrow R' \in (D[<a>])_c, c < b, R' \subseteq <a>, \overline{p} \notin <a> \Rightarrow \]

by \((2.11)\)

\[ p \in T_{\overline{D}[<a>]}^{k+1}, p \notin <a>. \]

We have thus proved that \(<a> \subseteq T^*(D[<a>]) = T^*(\overline{D}) \cup <a> = \langle a \rangle \).

To prove that the reverse of the last implication also holds, a few more considerations are necessary. Among the properties \( p \in T_{\overline{D}[<a>]}^{k+1} \), if \( p \notin <a> \) then \((2.24)\) is immediate. So the proof may be restricted to the case \( p \notin <a> \). By \((2.11)\)

\[ \exists p \leftarrow Q, Q' \in (D[<a>])_b, (Q, Q') \subseteq T_{\overline{D}[<a>]}^k, \neg \exists p \leftarrow \emptyset \in (D[<a>])_c, c < b. \]

Also \( \overline{p} \notin <a> \) otherwise no rule for \( p \) could belong to \( D[<a>] \). As \( Q \subseteq T_{\overline{D}[<a>]}^k \) then there must be rules in \( D[<a>] \) for each property in \( Q \) and so \( \overline{Q} \cap <a> = \emptyset \).

If \( p \in T_{\overline{D}[<a>]}^{k+1} \) using a rule in \( b \) then, beyond the inexistence of facts \( \overline{p} \) in \( c < b \), it is possible to conclude also, for at least one rule \( p \leftarrow Q, Q' \in (D[<a>])_b \) that

\[ \neg \exists p \leftarrow R' \in (D[<a>])_c, c < b, R' \subseteq <a> \]

because otherwise \( p \) would not belong to \( T^*(D[<a>] \), by \((2.11)\) and \((2.15)\), and so \( \overline{p} \in T^*(D[<a>] \) and, by \((2.14)\), \( p \in \Delta_{\overline{D}[<a>]} \) thus implying that \( \overline{p} \notin <a> \), which contradicts the existence of rules for \( p \) in \( D[<a>] \).

The extra properties generated by \( D[<a>] \), the HST without the explicitly overridden rules, are exactly those obtained from the residual theories \( \overline{D} \) after the division by the characterization. So \( T^*(D[<a>] \) contains both the characterization and the ambiguous residue.

The skeptical strategy contrasts with the credulous or brave approach of considering each of the possible alternatives in turn. Opting for \( p \) would give rise to a credulous characterization and choosing \( \overline{p} \) to another one.
Definition 66 (Credulous characterizations) Let $O$ be a consistent subset of the ambiguous residue $T^*(\overline{D})$. Then $<a>_O = S_{D/O} \uparrow \omega$ is a credulous characterization of $a$ if and only if $O$ is a minimal set such that

i) $O \subseteq <a>_O$

ii) $T^*(D[<a>_O]$ is consistent.

The set $O$ is required to be consistent because it represents the disambiguating options standing behind a particular characterization. The credulous characterization $<a>_O$ is the set of conclusions generated from the restricted theories obtained by overriding the rules for properties in $\overline{O}$. The requirement that the resulting credulous characterization must contain its own disambiguating options embodies a kind of stability. The conflicts mentioned in $O$ are resolved not by neutralization but by choosing one of the competitors. Then the bottom-up machinery of definition 49 is applied, regenerating $<a>$ and possibly some other properties. Some subsidiary conflicts, which were masked behind the main neutralizations, may now be resolved by overriding. Because of the requirement of consistency of $T^*(D[<a>_O)$, credulous characterizations have no neutralizations left.

Example 27 Credulous characterization.

![Diagram](image)

Figure 2.18: Overriding on top of neutralization.

The skeptical characterization for $a$ in figure 2.18 is empty.

\[ T^*(D^0) = \{r, \neg r, y, \neg y, p, \neg p\} \]
\[ \Delta_{D^0} = \emptyset \]
\[ <a> = \emptyset \]
There are only two credulous characterizations.

\[
O = \{r\} \\
T^*((D[O]^0)) = \{r, y, \neg y, p, \neg p\} \\
\Delta_{(D[O])^0} = \{r\} \\
(D[O])^1 = (D[O]^0)/\{r\} \\
T^*((D[O])^1) = \{\neg y, \neg p\} \\
\langle a \rangle_r = \{r, \neg y, \neg p\}
\]

Notice that the option for \(r\) created a situation where \(\neg y\) overrides \(y\) and \(p\) is no longer generated. As \(\{r\} \subseteq \langle a \rangle_r\), \(T^*(D[\langle a \rangle_r])\) is consistent and \(O = \{r\}\) is minimal then \(\langle a \rangle_r\) is a credulous characterization.

\[
O = \{\neg r\} \\
T^*((D[O]^0)) = \{\neg r, \neg p\} \\
\langle a \rangle_{\neg r} = \{\neg r, \neg p\}
\]

This is the other credulous characterization.

The set \(\langle a \rangle_{\neg y} = \emptyset\) is not a credulous characterization because it does not contain \(\neg y\) and \(T^*(D[\emptyset]) = \{r, \neg r, y, \neg y, p, \neg p\}\) is inconsistent.

The set \(\langle a \rangle_{r, y, p} = \{r, y, p\}\) is not a credulous characterization because it is not minimal. \(\square\)

**Proposition 67 (Safety)** The (skeptical) characterization is a subset of the intersection of all the credulous characterizations.

**Proof:**

This follows from definition 66 because the theories used to produce the credulous characterizations contain the same rules as the original HST except those for the properties in \(O\). They are expurgated, by the restriction operation, of the rules contradicting the options \(O \subseteq T^*(D)\) that, by corollary 62, are disjoint from \(\langle a \rangle\). So, the materials used in building the skeptical characterization remain untouched and the options taken about the unresolved conflicts can not affect them (see lemma 52).

\(\blacksquare\)

Notice that between the single skeptical and the several credulous characterizations it is possible to have semi-creduilus characterizations which correspond to keep the neutralization of some conflicts.
2.2 Rule inheritance

Corollary 68 (Credulous bound) Every credulous characterization is a subset of the weak characterization,

\[ <a>_O \subseteq <\overline{a}>. \]

Proof:

From proposition 67, \(<a> \subseteq <a>_O\). So the division of \(D\) by \(<a>\) is neutral with respect to the generation of the properties in \(<a>_O \setminus <a>\). They can as well be obtained from the residual theories \(\overline{D}\) restricted by \(O\), as \(O \subseteq T^*(\overline{D})\).

From (2.11), the only way a restriction can introduce new properties is by deleting an overriding property. In the residual theories there are no overridings as primary conflicts, though they may potentially exist on top of neutralizations, because any such property is included in the characterization. So, every property belonging to \(<a>_O\) is also in \(<a> \cup T^*(D/ <a>) = <\overline{a}>\). 

It is hard to conceive what kind of reasoning could be performed by arbitrarily picking one among the credulous characterizations, unless some external preference ordering is given in addition to specificity. Otherwise any conclusion of a HST could be deemed arbitrary.

The question can be raised of whether the (skeptical) characterization, which in principle is neutral with respect to ambiguities and contains only consensual properties, can be obtained through the intersection of all the credulous characterizations.

Example 28 Intersection and skepticism.

\[ T^*(D^0) = \{p, \neg p, r\} \]
\[ \Delta_{D^0} = T^*(D^0[T^*(D^0)]) = \emptyset \]
\[ <a> = \emptyset \]

Figure 2.19: Floating conclusion.

Skeptical characterization for target \(a\).

\[ T^*(D^0) = \{p, \neg p, r\} \]
\[ \Delta_{D^0} = T^*(D^0[T^*(D^0)]) = \emptyset \]
\[ <a> = \emptyset \]
Credulous characterizations. The index on \(<a>_p\) highlights the particular choice of ambiguous properties.

\[
<\!a\!>_p = \{p, r\}
\]
\[
<\!a\!>_{\neg p} = \{\neg p, r\}
\]
\[
<\!a\!>_p \cap <\!a\!>_{\neg p} = \{r\} \neq <\!a\!>
\]

Notice that figure 2.19 represents a counter-example for the conjecture intersection of all the credulous characterizations equals the skeptical characterization. The difference is the conclusion \(r\) which is common to both credulous characterizations but for reasons which vary from one to the other. The result is that the intersection contains \(r\) but does not provide any support for it, regarding the relevant theories. So \(r\) is a floating conclusion.

Example 28 displays the two reasons why it is not desirable to elect as the ideal semantics the intersection of all the credulous characterizations. First, it is not possible to explain the result gotten as the set of conclusions associated to a composed theory obtained by a simple operation of overriding. There may be conclusions whose supporting rules are all inactive. Second, to justify \(r\) it is needed to reason by cases to show that, no matter the choices made about the ambiguities, the conclusion is always present. The number of cases to consider is exponential on the number of ambiguities.

It is far nicer to have a direct approach to the construction of the (skeptical) characterization that avoids the computation of all the credulous characterizations, like the one presented before and based on neutralization. Specially if the result contains all the materials to support itself with respect to a (sub)set of the rules in the HST, therefore preventing the phenomenon of floating conclusions. It means enforcing the explicit information view even at the cost of missing some conclusions that would be unavoidable in a two-valued total setting.

The subject of ambiguity is much more subtle than it may look at first sight. The interaction of neutralizations and overdridings is not always intuitive.

**Example 29** Interacting ambiguities.

In figure 2.20, \(w\) stands for worker, \(s\) for student, \(a\) for adult, \(y\) for young, and \(m\) for married. With respect to the target \(ws\) (working student), there is an obvious ambiguity on property \(a\) which is resolved by neutralization. If we use the method of performing it immediately, then without \(a\) and \(\neg a\) the rule \(m \leftarrow a\) becomes inactive and \(m\) cannot be concluded, paving the way for \(y\) and \(\neg m\) to be inherited, given that they turn out to be not opposed.
This result is rather odd, as can be learned from credulous reasoning, if we suspend for a while our directly skeptical perspective. One credulous characterization will be \( \langle ws \rangle_{\neg a} = \{ \neg a, y, \neg m \} \). The other settles for \( a \) but it still faces another ambiguity, now on \( m \), which causes a second branching, \( \langle ws \rangle_{a,m} = \{ a, y, m \} \) and \( \langle ws \rangle_{a,\neg m} = \{ a, y, \neg m \} \). There is at least one credulous characterization containing \( m \) which confirms the existence of a second ambiguity, against the verdict in the first paragraph of this example.

The problem is that in the ambiguity \( a - \neg a \) both sides have equal strength but in the other case, while \( \neg m \) is also a conclusion with a non-contested support, \( m \) depends on the former ambiguity. So \( m - \neg m \) can be seen as a sort of second-order ambiguity with unbalanced sides. In more complex examples, the situation can be iterated and it may confront first-citizen alternatives with successively weaker opponents. No matter the relative strengths, there is always a certain ambiguity and a truly skeptical approach, in its quest for safe conclusions, will refuse to make a choice if there is the faintest doubt.

What enabled the initial assertion that \( \neg m \) is to be generated was the too early neutralization of \( a - \neg a \) that reduced both properties to nonexistence. The role played by the pair of revealing an ambiguity is thus blocked and restrained to a local effect. In a manifestation of opportunism \( \neg m \) profits from the lack of opposition to impose himself, though it is not a genuine indisputable conclusion.

The semantics of definition 49 does not perform neutralizations on a casuistic basis, rather it cancels all the conflicting properties at the same time. So, the effect of the pair \( a - \neg a \) remains long enough to generate \( m \), though in a previously condemned way, and neutralize \( \neg m \). This way the ambiguity is propagated and opportunism avoided, before that neutralization cleans up the result. The characterization of the class working student reduces to \( \langle ws \rangle = \{ y \} \). \( \square \)

Conclusions derived from a contradiction will not be part of the characterization but may be able to block the respective complementary properties. It is crucial to avoid opportunism but, in other situations, it may rule out properties which belong to all the credulous characterizations, as happened with the floating conclusions.
Example 30 *Contradiction supported rules.*

![Diagram](image)

Figure 2.21: Impossible conclusion.

The property \( \neg p \) is simultaneously supported by both \( r \) and \( \neg r \), a contradiction. After neutralization it is doomed to disappear and, further, it can not belong to any credulous characterization. Our directly skeptical method, in order to avoid reasoning by cases when propagating ambiguities, merges all the alternatives on a single pass, but does not prevent interferences among them. That is why \( \neg p \) is generated, despite its obvious impossibility. The bad result is that it neutralizes \( p \), an intuitively acceptable property, giving an empty characterization for \( a \). Anyway, these situations are not likely to occur in practice because there seems to be something wrong, from the knowledge representation viewpoint, in a rule that depends either directly or indirectly on a contradiction.

Example 30 is a case where the ambiguity is excessively propagated. Ironically, ambiguity blocking, i.e., preliminarily performing the obvious neutralization, would give the right answer. But still we prefer the former because, at most, it will miss some reasonable properties, while the latter may give dubious answers. Even if a compromise could be found it would probably mean an increase in computational complexity.

We have already seen that the directly skeptical characterization does not contain all the properties in the intersection of the credulous characterizations, for example the floating conclusions or properties neutralized by *impossible* conclusions. Proposition 67 says that it is always a subset of the common properties. Example 29 shows that an ambiguity blocking approach would give rise to opportunism and allow the characterization to surpass the intersection. As long as we stick to the ambiguity propagating version this problem will not occur.
2.3 Evaluation of the inheritance mechanisms

The chapter concludes with an evaluation, against the criteria of section 1.2.2, of the achieved results.

Predicate inheritance

1. The characterizations are consistent.

2. Contradictions between conclusions in a class and the respective defaults are resolved in favor of the local and against the inherited property. Contradictions between two inherited defaults conduct to neutralization, as well as local contradictions in the class conclusions. However, specificity is not always taken good care of because the local nature of predicate inheritance may misinterpret as requiring neutralization a situation that is actually of the overriding kind, when alternative paths exist between a pair of classes.

3. Exceptions should be kept to a minimum and generally they are, provided that ambiguity is resolved by neutralization. However, two features of predicate inheritance are at odds with this criterion. Rules which are not activated in a certain class, due to lack of support, may turn out to be not satisfied, if the support becomes available in a subclass, resulting in an implicit overriding of their heads. To remove a conflict between conclusions in a single class, defaults which are not explicitly contradicted may be overridden, in addition to the contradictory pair.

4. Properties in the characterization may be not supported. The characterization of a class is the least fixed-point of its local theory, which contains supported properties, extended with the non overridden defaults. The defaults are supported in the class where they are produced but may become unsupported in a subclass and still remain there, by inheritance.

Rule inheritance

1. The characterizations are consistent.

2. Overriding and neutralization are properly dealt with, relatively to the dynamic understanding of specificity, as applied to the construction of arguments, instead of statically classifying the rules.

3. Every conclusion which is not contradicted is inherited, and the overridden properties are the ones imposed by the previous criterion. So, exceptions are kept to a minimum, although it may be argued that,
in some cases, the propagation of ambiguities allows an "impossible" property, in the sense that it depends on two contradictory literals, to neutralize an otherwise legitimate conclusion.

4. As the characterization is the least model of a subset of the rules in the relevant theories, it only contains supported properties.
Chapter 3

Top-down inference

In the previous chapter we studied a bottom-up method for building the complete characterization of a target class. The definition based on fixed-points is good for precisely establishing the intended meaning of a HST, but is not practical from a computational viewpoint. The subject of the current chapter is a goal-directed computation which, following a top-down approach, is able to answer whether a given class enjoys or not a certain property.

3.1 The inference system

The goal-oriented procedure is defined through a set of inference rules. As properties are defeasible, in the sense that a subclass may be an exception and not verify a property true for one of its superclasses, every derivation in a HST must be referred to a target class. It will be called $a$.

3.1.1 Derivation relations

The inference system we designed (see figure 3.1) is based on three mutually dependent relations. The *strong derivation relation*, $\vdash_a p$, states that $p$ is indisputably a property for $a$. This means that the properties that $p$ depends on (the body of a rule for $p$) are also in the strong relation and that there is no evidence supporting the opposite $\overline{p}$ in $a$. The inference rules [SV], [SC] and [SR] define the empty goal, conjunction and reduction for the strong derivation relation.

There is evidence for $p$ in $a$ if $p$ belongs to the *weak derivation relation*, $\triangledown_a p$, i.e., if there is a line of reasoning leading to that conclusion even if some of the intervening properties, including the conclusion, happen to be contested by an alternative line of reasoning. Thus $\triangledown_a p$ and $\triangledown_a \neg p$ may both occur revealing an ambiguity which gives rise to neutralization as neither $p$ nor $\neg p$ may then belong to
\[ [SV] \quad \vdash_{a} \emptyset \]

\[ [SC] \quad \vdash_{a} p \quad \vdash_{a} Q \quad \vdash_{a} p, Q \]

\[ [SR] \quad \vdash_{a} Q \quad \forall_{a} \overline{p} \quad \{ \ p \leftarrow Q \in \mathcal{D}_{b} \ \} \]
\[ \vdash_{a} p \quad a \leq b \]

\[ [WV] \quad \neg \vdash_{a} \emptyset \]

\[ [WC] \quad \vdash_{a} p \quad \vdash_{a} Q \quad \vdash_{a} p, Q \]

\[ [WR] \quad \neg \vdash_{a} Q \quad \forall_{a} \overline{p} \quad \{ \ p \leftarrow Q \in \mathcal{D}_{b} \ \} \]
\[ \vdash_{a} p \quad a \leq b \]

\[ [LO] \quad \vdash_{a} Q \quad \{ \ p \leftarrow Q \in \mathcal{D}_{c} \ \} \]
\[ \vdash_{b} p \quad a \leq c < b \]

Figure 3.1: Inference rules.
the strong relation. The weak derivation relation has a form similar to the previous one (see [WV], [WC] and [WR]) and is true of the properties which have a reason for being true, no matter whether a neutralizing property is also (weakly) derived or not. The weak relation is responsible for propagating ambiguities and its role is to inhibit the strong derivation whenever it relies on ambiguous literals.

To cope with overriding, the weak relation itself must be restricted (see “restricted skepticism” in [HTT90] for a similar concept), i.e., it must not be contradicted by more specific information. Property $p$ belongs to the weak relation if it is the head of a rule in the definition for a certain class $b$, all the properties in the respective body are also in the weak relation, and it is not overridden.

The local derivation relation $\vdash^b_a p$ is true if, in a path between $a$ and $b$, there is a rule for $p$ whose body is in the strong relation for $a$. Strong derivation is required for the bodies because only secure information can be allowed to eliminate, by overriding, a doubt expressed by the weak relation. As will be seen later, this is the operational counterpart of the restriction of overriding just to facts. The local derivation relation (see [LO]) has two parameters: the target class and the class where the definition to be overridden lies. An overriding rule must belong to a class definition in between.

Given that the local relation is based on the strong relation, our definition may seem cyclic. The relation $\vdash_a p$ depends on $\vdash_a \overline{p}$ which depends on $\vdash^b_a p$. However, the second invocation of $p$ is done in a restricted context and does not recur. Cycles may still show up but only due to self-supported property rules such as $p \leftarrow p$ or $p \leftarrow \neg p$. The restriction in the search space when checking for contradictions is due to the fact that one is not trying to prove the complement in general (that would be symmetrical, and lead to ping-pong situations) but focused in the starting rule class.

### 3.1.2 Derivations

A derivation is a set of derivation trees with a distinguished main tree plus a marking map.

In a derivation tree, the nodes are labeled by goals and the edges, marked by horizontal lines, correspond to the application of inference rules. As these deal with three different relations, the goals $\alpha Q$ in derivations, where $\alpha$ is one of $\vdash$, $\vdash^\top$, $\vdash^\neg$, resp. for the strong, weak and local derivations, must explicitly mention the relation symbol along with $Q$, a set of properties. These are the positive goals. Their opposites, the negative goals, are for the negated relations $\vdash^\neg$, $\vdash^\neg^\top$, $\vdash^\neg^\neg$. The children of a node are the subgoals introduced by the corresponding rule. The root of the tree is the main goal, or query, to be proved. The inner nodes are followed by reduction ([SR], [WR], [LO]) or conjunction ([SC],
[WC] rules. The leaf nodes are of three kinds: empty goal, positive goal and negative goal.

There is a marking map defined for leaf nodes. A leaf that is marked successful, resp. failed, is followed by a horizontal line, resp. a cross. A marking is consistent if and only if it assigns the same mark to the leaves labeled by the same goal.

A derivation tree is:

- **infinite** if at least one of its branches has an infinite number of nodes, otherwise it is finite;
- **successful** if it is finite and all its leaves are marked successful;
- **failed** if it is infinite or at least one of its leaves is marked failed.

**Definition 69 (Derivation)** A set of derivation trees, one of which is called the main tree, plus a marking map is a derivation of the goal $\prec p$ if and only if

- The root of the main tree is labeled by $\prec p$.
- The root of each tree, except for the main tree, is the opposite of a negative goal in some tree in the derivation.
- For each leaf in a derivation tree labeled by a
  - empty goal - it is marked successful (for the application of [SV] or [WV]);
  - positive goal - there are no applicable rules and the node is marked failed;
  - negative goal -
    a) it is marked failed and the only tree rooted by the opposite goal is marked successful;
    b) it is marked successful and all the possible alternative derivation trees rooted by the opposite goal, considering every applicable rule, belong to the derivation and are marked failed.
- The marking is consistent.

When $\not\prec p$ succeeds because there are no rules for $\prec p$, we usually do not depict the corresponding single failed tree.

A derivation is classified as finite, successful, failed if and only if its main tree is finite, successful, failed. A derivation is cyclic if it has a subderivation whose main goal is the opposite of a negative goal in that subderivation.

The definition of derivation is of an inductive nature. If a derivation loops this induction may get into trouble. A positive loop produces in a tree an infinite
branch, which therefore has no leaf and no marking. A negative loop makes the marking of a negative goal depend on itself. So it is possible to have different consistent markings. We stress that a set of trees corresponds to as many derivations as the different markings it complies with.

So, a derivation contains the main derivation tree for the main goal as well as a set of (sub)derivation trees required to justify the marking of the negative goals. Notice that infinite (thus failed) subderivations may occur in a finite derivation.

Analyzing the inference rules, negative goals may arise in two situations. In its weak form, in a strong derivation, and in the local form, in weak derivations. They are understood as a check for the failure of a derivation of the opposite positive goal.

The derivation trees may be arranged in levels. For instance in a strong derivation, the top one contains the main (strong) tree, the first sublevel contains the weak subderivations and the second the local ones. As a local derivation is converted in a strong derivation immediately after the first step, it may be considered a special case of the latter. The third level contains again weak derivations, etc.

**Example 31 A sample derivation.**

![Derivation Diagram](image)

Figure 3.2: A derivation using the cascaded neutralization.

The derivation of figure 3.2 is for the HST of example 29.

The strong derivation of \( \neg m \) fails, though the strong derivation of the body of the rule \( \neg m \leftarrow y \) succeeds, because a weak derivation of \( m \) succeeds, meaning that there is some evidence for \( m \) (see the right half of the figure). In this case, the evidence is strictly weak since it can be checked that there is no successful strong derivation of \( m \) too, because the opposite \( \neg m \) forces neutralization and because it depends on a neutralized property \( a \), which is already just weak. This result agrees with the skeptical ambiguity propagating view already mentioned in the fixed-point construct. Notice that besides the weak derivation of \( m \) some other conclusions about the class \( ws \) were produced. The weak derivation of property
\[ \vdash_{ws} = \{ y \} \]
\[ \vdash_{ws} = \{ y, a, \neg a, m, \neg m \} \]
\[ \vdash_{ws} = \emptyset, \quad \forall x \in C. \] (3.1)

The local derivation \( \vdash^s_{ws} y \) fails because there is no definition for \( y \) below \( s \). \( \square \)

The set of rules in figure 3.1 deals with exceptions of exceptions. A strong derivation of \( p \) with target class \( a \) checks for the failure of all the weak derivation of \( p \), catering for both neutralization and overriding situations. The rule for \( p \) may come from any of the target's superclasses, say \( b \), although only those not above the class containing the rule for \( p \), say \( c \), may lead to its defeat. If \( a \leq b < c \) it is overriding; if \( c \not< b \) and \( b \not< c \) it is neutralization. Nevertheless the weak derivation of \( p \) is subject to the failure of all the local derivations of \( p \), now restricted to start in a class \( d \) such that \( a \leq d < b \), i.e., just checking for overriding.

A set of inference rules is understood as defining the least relations that satisfy the rules. The rules in figure 3.1 mention simultaneously three relations. So the minimization must be performed on the triples \( I = (\vdash_{\neg a}, \vdash_a, (\vdash_{\neg a})_{a \leq z}) \), where \((\vdash_{\neg a})_{a \leq z}\) is a family of local derivation relations.

The requirement of satisfaction is usually enough to constrain the meaningful relations because minimality tackles the problem of eliminating arbitrary extra properties, which are not justified by the rules but nevertheless satisfy them. In HST, as the relations are mutually defined through negation, it may happen that when one becomes smaller another one gets bigger. So, the least triple, in a set inclusion sense, may not even exist. The presence of unjustified extra properties may free otherwise required properties to belong or not to the relations, if a mere criterion of satisfiability is used. Notice that in particular the triple \( I = (\emptyset, \mathcal{P}, (\mathcal{P}, \mathcal{P}, \ldots)) \) always satisfies the inference rules. We add a further condition of derivability to eliminate unjustified properties in the derivation relations.

**Definition 70 (Triple)** A triple \( I = (\vdash_{\neg a}, \vdash_a, (\vdash_{\neg a})_{a \leq z}) \) of derivation relations is induced by the inference rules if and only if

- \( I \) satisfies the rules, and

- there exist successful derivations of each \( \alpha_a p \) such that \( p \in \alpha_a \) whose markings are mutually consistent.

According to this definition, (3.1) is the sole triple induced by the inference rules with respect to the HST of example 29.
3.1.3 Cyclic derivations

In this section we introduce the problem of derivations with loops across negative goals. They are distinct from infinite derivations, which contain loops just with positive goals. A negative loop does not produce an infinite branch and the number of trees in the derivation is finite. However a computation trapped in it would perform an infinite number of traversals of derivation trees. That’s why we called such derivations cyclic. Their distinguishing characteristic is that they contain a subderivation whose main goal is referred by a negative goal in the subderivation itself. Thus there may be different consistent markings of those goals. Whether a cyclic derivation succeeds or not depends on the marking of its main tree.

Example 32 Alternative relations.

\[
\begin{array}{c}
\begin{array}{c}
\neg p \\
a \vdash \neg p
\end{array} \\
\begin{array}{c}
b \\
a \vdash \neg p
\end{array}
\end{array}
\]

\[
\begin{array}{c}
\vdash_a \neg p \\
\vdash_a p \\
\vdash_a \neg p
\end{array}
\]

\[
\begin{array}{c}
\vdash_a \neg p \\
\vdash_a p \\
\vdash_a \neg p
\end{array}
\]

\[
\begin{array}{c}
\vdash_a \neg p \\
\vdash_a p \\
\vdash_a \neg p
\end{array}
\]

Figure 3.3: Auto-overriding.

During the derivation in figure 3.3 a loop is found, on the goal \(\not\vdash_a p\). This means that the derivation is cyclic.

The problem is that two different markings of the goals with a question mark can be consistently done, depending on whether the loop is interpreted as failure of the local/strong derivations or of the weak derivation. We represent in figure 3.4 a success by \(t\) and a failure by \(f\).

<table>
<thead>
<tr>
<th></th>
<th>(\vdash_a \neg p)</th>
<th>(\vdash_a p)</th>
<th>(\vdash_a \neg p)</th>
<th>(\vdash_a^b p)</th>
<th>(\vdash_a p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(I_1)</td>
<td>(t)</td>
<td>(f)</td>
<td>(f)</td>
<td>(t)</td>
<td>(t)</td>
</tr>
<tr>
<td>(I_2)</td>
<td>(f)</td>
<td>(t)</td>
<td>(t)</td>
<td>(f)</td>
<td>(f)</td>
</tr>
</tbody>
</table>

Figure 3.4: Two markings.
If $\vdash_a \neg p = f$ then the local goal $\vdash_a ^b p = f$ and thus $\not\vdash_a ^b p = t$. This implies that the weak goal $\vdash_a p$ succeeds and inhibits the strong goal $\vdash_a \neg p = f$, reinforcing our starting point. If, on the contrary, we think that the weak goal is the one which fails, $\vdash_a p = f$, then nothing prevents the local $\vdash_a ^b p = t$ from succeeding and overriding $\vdash_a \neg p$, which makes $\vdash_a p$ fail, confirming the conjecture.

Of course, the marking $I_1$ is rather nonsensical because it allows the strong derivation of $\neg p$ while at the same time denying its weak derivation. Moreover it is contradictory because $\vdash_a p$ also succeeds. When going from $I_1$ to $I_2$, $\vdash_a$ gets smaller but $\vdash_a$ increases. □

To give some insight on the behavior of the inference procedure, we show in proposition 71 that an odd situation like $p$ being strongly concluded but not weakly, must be based on a similar pair. Let us call such a pair $(\vdash p, \not\vdash p)$ a clash. For a finite HST, this means that a loop of clashes must exist.

**Proposition 71 (Loop of clashes)** If, with respect to a property $p$, we have simultaneously $\vdash p$ and $\not\vdash p$ then $\vdash p$ depends on a property $r$ such that $\vdash r$ and $\not\vdash r$, either through the body of the rule supporting it or via overriding.

**Proof:**

Let us assume that there is a clash on $p$. If there is a successful derivation of $\vdash p$ using rule

$$p \leftarrow Q \in D_b$$  \hspace{1cm} (3.2)

then, by [SR], there are successful strong derivations for each $q_i \in Q$ and $\not\vdash \bar{p}$. As, by assumption, $\not\vdash p$ succeeds, we have that, by [WR], for all rules $p \leftarrow Q$, including (3.2),

$$\not\vdash Q \lor \vdash \bar{p}.$$ 

a) If there is a $q_i \in Q$ such that $\not\vdash q_i$ then $r = q_i$, given that $\vdash q_i$ succeeds.

b) If $\vdash \bar{p}$ (the other way of failing the derivation of $\vdash p$) then there is a rule $\bar{p} \leftarrow S$ whose body is strongly derived. As, by hypothesis, $\not\vdash p$, then, by [SR], it follows that $\vdash \bar{p}$. From (3.2) we have $\not\vdash \bar{p}$, so $r = \bar{p}$.

As one may choose rule (3.2) to be any rule strongly deriving $p$, the proposition is proved. ■

So, it becomes apparent that the properties involved in a loop of clashes are related through the body of a rule or by overriding. If there were just dependencies of the first kind then the goals in the loop would be all positive (infinite derivation) and would belong neither to the strong nor to the weak relations. So, at least one
of the dependencies must be due to an overriding. But then, by case b) in the proof of proposition 71, we have both \( \vdash p \) and \( \vdash \neg p \) and the strong relation is contradictory.

Definition 70 does not completely settle the meaning of the set of inference rules in figure 3.1 because example 32 shows that there may not be a single smallest (in the subset inclusion sense) set of relations induced by the rules. The problem here is what is meant by \textit{smallest}.

Before characterizing exactly the defined relations, we introduce a translation from HSTs to logic programs that will help in setting up that characterization.

### 3.2 Translation to Logic Programming

To prove that the inference system is consistent, i.e., if \( \vdash_a p \) then \( \not\vdash_a \neg p \), we will use some results of the theory of Logic Programming. Some familiarity is assumed with the notions of stable model semantics [GL88], well-founded semantics [VGRS90] and extended stable model semantics [PP90].

#### 3.2.1 Direct translation

The following definition establishes the bridge between a HST and a logic program.

**Definition 72 (From HST to Logic Programming)** Let \( \mathcal{H} \) be a HST. \( \mathcal{H}^* \) is the corresponding logic program if and only if, with respect to each class \( a \) and for each rule \( p \leftarrow q_1, \ldots, q_n \) in \( D_c, a \leq c \), \( \mathcal{H}^* \) contains the following clauses:

\[
\begin{align*}
\vdash_a p & \leftarrow \vdash_a q_1, \ldots, \vdash_a q_n, \sim \vdash_a \neg p \\
\sim_a p & \leftarrow \sim_a q_1, \ldots, \sim_a q_n, \sim \vdash_a \neg p \\
\vdash^b_a p & \leftarrow \vdash_a q_1, \ldots, \vdash_a q_n, \forall b \geq c.
\end{align*}
\]  

(3.3)

In \( \mathcal{H}^* \), there are three kinds of propositional symbols, \( \vdash_a p \), \( \sim_a p \) and \( \vdash^b_a p \), for each property \( p \) mentioned in a rule head. The symbol \( \sim \) stands for the default negation. So, \( \mathcal{H}^* \) is a general logic program. Also, \( \mathcal{H}^* \) can be partitioned in independent subprograms for each target class. In the sequel, we will restrict the study to a designated target class and omit it from the symbols, for readability.

Each rule in the HST corresponds, in one of those partitions, to a single rule for a strong propositional symbol, \( \vdash p \), and another for the corresponding weak propositional symbol, \( \sim p \), in accordance to the global character of the corresponding inference relations. The overriding effect of a rule in the HST ranges over the
respective superclasses, excluding the class where it belongs. This behavior is modeled by the local propositional symbols, \( \vdash^b p \), which have the class \( b \), subject to overriding, as an extra parameter. A rule in a class \( c \) in \( \mathcal{H} \) is thus translated into as many rules for the local symbols with \( p \) as there are superclasses above \( c \), possibly none.

### 3.2.2 Transformational approach

The logic program \( \mathcal{H}^* \) is also the result of a sequence of transformational steps applied to the HST \( \mathcal{H} \), which preserve its original meaning.

The set of inference rules in figure 3.1 can be seen as a specification of a meta-interpreter for HST. Figure 3.5 contains a Logic Programming version of that meta-interpreter. As the inference rules simultaneously specify three different relations, these must be explicitly named in the meta-interpreter. There is the two-place strong predicate \( \vdash_A P \) that accepts a class \( A \) and a property \( P \) as arguments; the two-place weak predicate \( \vdash_{\neg A} P \), with the same arguments; and the local predicate \( \vdash\neg_A P \) with an extra argument for an auxiliary class.

There are three metapredicates to access the information contained in the HST: \( \text{rule}(B, P \leftarrow Q) \) is true if and only if there is a rule \( P \leftarrow Q \) in class \( B \); \( A \leq B \) and \( A < B \) involve a little processing over the basic isa relation \( \prec \). \( \overline{P} \) is the complementary property of \( P \).

\[
\begin{align*}
\vdash_A \emptyset \\
\vdash_A (P, Q) & \leftarrow \vdash_A P, \vdash_A Q \\
\vdash_A P & \leftarrow A \leq B, \text{rule}(B, P \leftarrow Q), \neg \vdash_{\neg A} \overline{P}, \vdash_A Q \\
\vdash_{\neg A} \emptyset \\
\vdash_{\neg A} (P, Q) & \leftarrow \vdash_{\neg A} P, \vdash_{\neg A} Q \\
\vdash_{\neg A} P & \leftarrow A \leq B, \text{rule}(B, P \leftarrow Q), \neg \vdash\neg_A \overline{P}, \vdash_{\neg A} Q \\
\vdash\neg_A P & \leftarrow A \leq C, C < B, \text{rule}(C, P \leftarrow Q), \vdash_A Q
\end{align*}
\]

Figure 3.5: Simplified version of the meta-interpreter.

The similarity between the inference rules and the meta-interpreter allows us to see the latter as an operationally correct implementation of the former. The main problem resides in the interpretation of negation. Notice that the references to the nonderivability of certain properties are modeled by the default negation implying that a compatible meaning be assigned to this connective. The well-founded semantics [VGRS90] is a better approximation to the notion of nonderivability than
the finite failure [Cla78] interpretation of negation. We will return to this problem later.

The meta-interpreter plus a representation of the HST $\mathcal{H}$ in terms of $\text{rule}/3$, $\leq/2$ and $</2$ is a means of disclosing the information conveyed by $\mathcal{H}$, in the sense given by the inference rules. The two components can be combined together in order to specialize the meta-interpreter by partially evaluating it [LS91, LS88] with respect to $\mathcal{H}$. Such partial evaluation is mainly concerned with replacing each call to a metapredicate by its possible occurrences. So each rule in $\mathcal{H}$ will give rise to three clauses in the specialized meta-interpreter, as many as the existing calls to $\text{rule}/3$, all of them with the arguments $P, Q$ and the source class (either $B$ or $C$) instantiated. In the end, the predicate $\text{rule}/3$ becomes completely evaluated and disappears. In this step, we also perform the complete evaluation of the complementation $\overline{P}$ and the partial evaluation, just for the cases of conjunction and empty body, of the strong and weak relations.

You may notice in the example 33 that at this point the specialized program is a translation of the original HST with an interesting property: it is modular, i.e., adding a new rule to a class in the HST (or even to a new class) means adding three clauses to the translation (plus the hierarchic information), whose shape is easily related to the original one. The other rules remain unchanged. In order to run, the translation requires a few simple definitions to manipulate the hierarchy.

The specialization of the meta-interpreter may be pushed further. The second step is the partial evaluation of the metapredicates $\leq/2$ and $</2$, which leads to the program's full instantiation. The relationship with the original $\mathcal{H}$ is less systematic because the number of rules generated depends heavily on its position in the hierarchy. It is also less modular because adding a new class accommodating a new rule may require changes to the translation of other rules besides the new one.

Example 33 Specializing the meta-interpreter.

![Diagram](image)

Figure 3.6: The swimming cetaceous, again.
Step 0 Meta-interpreter of figure 3.5 plus HST.

| rule(animal, ¬swim ← ø) | cetaceous<mammal | A ≤ A |
| rule(mammal, milk ← ø) | cetaceous<aquatic | A ≤ B ← A < B |
| rule(aquatic, swim ← fin) | mammal<animal | A < B ← A < B |
| rule(aquatic, fin ← ø) | aquatic<animal | A < B ← A < C, C < B |
| rule(aquatic, ¬milk ← ø) |

\[
\begin{align*}
\vdash_A \emptyset \\
\vdash_A (P, Q) & \leftarrow \vdash_A P, \vdash_A Q \\
\vdash_A P & \leftarrow A \leq B, \text{rule}(B, P ← Q), \sim\vdash_A \overline{P}, \vdash_A Q \\
\vdash_A \emptyset \\
\vdash_A (P, Q) & \leftarrow \vdash_A P, \vdash_A Q \\
\vdash_A P & \leftarrow A \leq B, \text{rule}(B, P ← Q), \sim\vdash_A \overline{P}, \vdash_A Q \\
\vdash_A P & \leftarrow A < C, C < B, \text{rule}(C, P ← Q), \vdash_A Q
\end{align*}
\]

Step 1 Partial evaluation of rule/3, complementation, conjunction and empty body.

| cetaceous<mammal | A ≤ A |
| cetaceous<aquatic | A ≤ B ← A < B |
| mammal<animal | A < B ← A < B |
| aquatic<animal | A < B ← A < C, C < B |

\[
\begin{align*}
\vdash_A \neg\text{swim} & \leftarrow A \leq \text{animal}, \sim\vdash_A \text{swim} \\
\vdash_A \neg\text{swim} & \leftarrow A \leq \text{animal}, \sim\vdash_A \neg\text{animal} \text{ swim} \\
\vdash_A \neg\text{swim} & \leftarrow A \leq \text{animal}, \text{ animal} < B \\
\vdash_A \text{milk} & \leftarrow A \leq \text{mammal}, \sim\vdash_A \neg\text{milk} \\
\vdash_A \text{milk} & \leftarrow A \leq \text{mammal}, \sim\vdash_A \neg\text{mammal} \neg\text{milk} \\
\vdash_A \text{milk} & \leftarrow A \leq \text{mammal}, \text{ mammal} < B \\
\vdash_A \text{swim} & \leftarrow A \leq \text{aquatic}, \sim\vdash_A \neg\text{swim}, \vdash_A \text{fin} \\
\vdash_A \text{swim} & \leftarrow A \leq \text{aquatic}, \sim\vdash_A \neg\text{aquatic} \neg\text{swim}, \vdash_A \text{fin} \\
\vdash_A \text{swim} & \leftarrow A \leq \text{aquatic}, \text{ aquatic} < B, \vdash_A \text{fin} \\
\vdash_A \text{fin} & \leftarrow A \leq \text{aquatic}, \sim\vdash_A \neg\text{fin} \\
\vdash_A \text{fin} & \leftarrow A \leq \text{aquatic}, \sim\vdash_A \neg\text{aquatic} \neg\text{fin} \\
\vdash_A \text{fin} & \leftarrow A \leq \text{aquatic}, \text{ aquatic} < B \\
\vdash_A \neg\text{milk} & \leftarrow A \leq \text{aquatic}, \sim\vdash_A \text{milk} \\
\vdash_A \neg\text{milk} & \leftarrow A \leq \text{aquatic}, \sim\vdash_A \neg\text{aquatic} \text{ milk} \\
\vdash_A \neg\text{milk} & \leftarrow A \leq \text{aquatic}, \text{ aquatic} < B
\end{align*}
\]
Step 2 Partial evaluation of ≤ /2 and < /2.

\[\vdash \text{animal} \rightarrow \text{swim} \leftarrow \sim \vdash \text{animal} \rightarrow \text{swim}\]

\[\vdash \sim \text{animal} \rightarrow \text{swim} \leftarrow \sim \vdash \text{animal} \rightarrow \text{swim}\]

\[\vdash \text{mammal} \rightarrow \text{swim} \leftarrow \sim \vdash \text{mammal} \rightarrow \text{swim}\]

\[\vdash \sim \text{mammal} \rightarrow \text{swim} \leftarrow \sim \vdash \text{mammal} \rightarrow \text{swim}\]

\[\vdash \text{mammal} \rightarrow \text{milk} \leftarrow \sim \vdash \text{mammal} \rightarrow \sim \text{milk}\]

\[\vdash \sim \text{mammal} \rightarrow \text{milk} \leftarrow \sim \vdash \text{mammal} \rightarrow \sim \text{milk}\]

\[\vdash \text{animal} \rightarrow \text{milk}\]

\[\vdash \text{mammal} \rightarrow \text{milk}\]

\[\vdash \text{aquatic} \rightarrow \text{swim} \leftarrow \sim \vdash \text{aquatic} \rightarrow \text{swim}\]

\[\vdash \sim \text{aquatic} \rightarrow \text{swim} \leftarrow \sim \vdash \text{aquatic} \rightarrow \text{swim}\]

\[\vdash \text{aquatic} \rightarrow \text{swim} \leftarrow \sim \vdash \text{aquatic} \rightarrow \sim \text{swim}, \vdash \text{aquatic} \rightarrow \text{fin}\]

\[\vdash \sim \text{aquatic} \rightarrow \text{swim} \leftarrow \sim \vdash \text{aquatic} \rightarrow \sim \text{swim}, \vdash \sim \text{aquatic} \rightarrow \text{fin}\]

\[\vdash \text{animal} \rightarrow \text{aquatic} \rightarrow \text{swim}\]

\[\vdash \text{aquatic} \rightarrow \text{fin}\]

\[\vdash \text{aquatic} \rightarrow \sim \text{fin}\]

\[\vdash \text{aquatic} \rightarrow \text{fin}\]

\[\vdash \sim \text{aquatic} \rightarrow \text{fin}\]

\[\vdash \text{animal} \rightarrow \text{aquatic} \rightarrow \sim \text{fin}\]

\[\vdash \text{aquatic} \rightarrow \sim \text{milk}\]

\[\vdash \sim \text{aquatic} \rightarrow \text{milk}\]

\[\vdash \text{animal} \rightarrow \text{aquatic} \rightarrow \sim \text{milk}\]

\[\vdash \text{cetaceous} \rightarrow \text{swim} \leftarrow \sim \vdash \text{cetaceous} \rightarrow \text{swim}\]

\[\vdash \sim \text{cetaceous} \rightarrow \text{swim} \leftarrow \sim \vdash \text{cetaceous} \rightarrow \text{swim}\]

\[\vdash \text{cetaceous} \rightarrow \text{swim} \leftarrow \sim \vdash \text{cetaceous} \rightarrow \sim \text{swim}, \vdash \sim \text{cetaceous} \rightarrow \text{fin}\]

\[\vdash \sim \text{cetaceous} \rightarrow \text{swim} \leftarrow \sim \vdash \text{cetaceous} \rightarrow \sim \text{swim}, \vdash \sim \text{cetaceous} \rightarrow \text{fin}\]

\[\vdash \text{animal} \rightarrow \text{cetaceous} \rightarrow \text{swim}\]

\[\vdash \text{cetaceous} \rightarrow \text{fin}\]

\[\vdash \text{cetaceous} \rightarrow \sim \text{fin}\]

\[\vdash \text{cetaceous} \rightarrow \text{fin}\]

\[\vdash \sim \text{cetaceous} \rightarrow \text{fin}\]

\[\vdash \text{animal} \rightarrow \text{cetaceous} \rightarrow \text{fin}\]

\[\vdash \text{cetaceous} \rightarrow \sim \text{milk}\]

\[\vdash \sim \text{cetaceous} \rightarrow \text{milk}\]

\[\vdash \text{animal} \rightarrow \text{cetaceous} \rightarrow \sim \text{milk}\]
Figure 3.7: Well-founded semantics of the program in step 2.

Step 2 shows a general logic program, or better, a collection of general logic programs, one for each class in the HST. Each of them is independent from the others. The corresponding well-founded semantics is represented in the tables of figure 3.7, where √ means a true literal, ~a false literal and blank entries correspond to literals that are not mentioned in the program. Of course, only the first line in each table contains strong conclusions and thus has a correspondence with the characterization. The rest may be seen as auxiliary stepping stones.

As a final cosmetic step we may freeze each instantiated atom by transforming it into a propositional symbol containing the corresponding derivation relation, property, target class and, if applicable, overridden class. This form is the one obtained by the direct translation process introduced in definition 72.

The purpose of the previously described transformational process is twofold. First, it is intended to convince the reader of the close relationship between the operational semantics of the HST $\mathcal{H}$ and its translation $\mathcal{H}^*$, so as to justify the extension to the former of some results obtained for the latter. Besides that, it
provides a naive implementation of that operational semantics, which may serve as a basis for further refinements and optimizations. See Annex A for an actually running Prolog version of the meta-interpreter.

3.3 Proving the consistency

3.3.1 In the translation

Let $\tilde{B}$ be the set of all the propositional symbols in $\mathcal{H}^*$ and their default negations. We also use the abbreviation $\nvdash p$ for the default negation $\neg\vdash p$, and write similarly $\nvdash^c p$ and $\nvdash^b p$ for $\neg\vdash p$ and $\neg\vdash^b p$, respectively. The well-founded model [VGRS90] for $\mathcal{H}^*$ is the least fixed-point of the mapping

$$W : 2^B \rightarrow 2^B$$

given by

$$W(I) = T(I) \cup \sim U(I)$$

(3.4)

where $T$ is the immediate consequence operator and $U$ is the unfounded set operator. (The notation $\sim U(I)$ represents the set of all $\sim A$ such that $A \epsilon U(I)$.) As $\mathcal{H}$ is finite and the translation ascribes a finite number of clauses to each HST rule, there is a $n$ such that the well-founded model of $\mathcal{H}^*$ is $W \uparrow n$. For a given literal in $\tilde{B}$, we say it is true, false or undefined if it is respectively true, false or undefined in the well-founded model of $\mathcal{H}^*$.

Theorem 73 (Strong-weak soundness) For any property $p$ in $\mathcal{H}$, if $\vdash p$ is true with respect to $\mathcal{H}^*$ then $\nvdash p$ is either true or undefined with respect to $\mathcal{H}^*$.

Proof:

This proof is mainly due to Luís Monteiro. Let us assume that $\vdash p$ is true and $\nvdash p$ is false (that is $\nvdash^c p$ is true) for some $p$. The pair $(\vdash, \nvdash^c)$ is a clash. Denoting by $\mathcal{M}$ the well-founded model of $\mathcal{H}^*$, we must show that $\mathcal{M}$ is free of clashes. In particular, we show that the assumption that $\mathcal{M}$ has a clash leads to a contradiction, which establishes the desired result.

If $A$ denotes any of the propositions $\vdash p$, $\nvdash p$, $\vdash^c p$, $\vdash^b p$ and $\nvdash^b p$, let the rank $\rho(A)$ of $A$ be the least integer $n$ such that $A \epsilon W \uparrow n$, provided that $A$ is not undefined in $\mathcal{M}$. The rank $\rho(\vdash p, \nvdash^c p)$ of a clash $(\vdash p, \nvdash^c p)$ is defined to be $\min\{\rho(\vdash p), \rho(\nvdash^c p)\}$.

Assuming $\mathcal{M}$ has a clash, pick a clash of least rank. We distinguish two cases:

1. $\rho(\vdash p, \nvdash^c p) = \rho(\vdash p)$;
II. $\rho(\vdash p, \not\models p) = \rho(\not\models p)$.

We now consider each of the two cases in turn.

(I) Let $\rho(\vdash p) = n + 1$ for some $n \geq 0$, so that $\vdash p \in T(W \uparrow n)$. There is a rule

$$p \leftarrow q_1, \ldots, q_m$$

in some class $b$ in $H$ such that

$$\vdash q_1, \ldots, \vdash q_m, \not\models \bar{p} \in W \uparrow n.$$  \hspace{1cm} (3.5)

We also have the clause

$$\not\models p \leftarrow \not\models q_1, \ldots, \not\models q_m, \not\models^b \bar{p}$$

in $H^*$. Since $(\vdash p, \not\models p)$ is a clash, we have $\not\models p \in M$. It follows that either $\not\models q_i \in M$ for some $i = 1, \ldots, m$ or $\vdash^b \bar{p} \in M$. In the first case we would have a clash $(\vdash q_i, \not\models q_i)$, by (3.5), and

$$\rho(\vdash q_i, \not\models q_i) \leq \rho(\vdash q_i) \leq n < \rho(\vdash p) = \rho(\vdash p, \not\models p),$$

contradicting the choice of $(\vdash p, \not\models p)$ as a clash of least rank. So we must have $\vdash^b \bar{p} \in M$. This means that there is a class $c < b$ in $H$ and a rule

$$\bar{p} \leftarrow r_1, \ldots, r_k$$

in $c$ such that

$$\vdash r_1, \ldots, \vdash r_k \in M.$$  \hspace{1cm} (3.6)

In $H^*$ we also have the clause

$$\vdash \bar{p} \leftarrow \vdash r_1, \ldots, \vdash r_k, \not\models p,$$

and since $\not\models p \in M$ by hypothesis, we conclude by (3.6) that $\vdash \bar{p} \in M$. As $\not\models \bar{p} \in W \uparrow n \subseteq M$, by (3.5), we have a clash $(\vdash \bar{p}, \not\models \bar{p})$ and

$$\rho(\vdash \bar{p}, \not\models \bar{p}) \leq \rho(\not\models \bar{p}) \leq n < \rho(\vdash p) = \rho(\vdash p, \not\models p),$$

contradicting once again the minimality of the rank of $(\vdash p, \not\models p)$.

(II) We consider now the case where $\rho(\vdash p, \not\models p) = \rho(\not\models p)$. By case (I), we may assume that any clash $(\vdash q, \not\models q)$ such that $\rho(\vdash q, \not\models q) = \rho(\vdash p, \not\models p)$ satisfies

$$\rho(\vdash q, \not\models q) = \rho(\not\models q).$$  \hspace{1cm} (3.7)

Furthermore, among all such clashes we may assume that $(\vdash p, \not\models p)$ has been chosen so that $\rho(\vdash p)$ has the least possible value.
Let $\rho(\not\vdash p) = n + 1$ for some $n \geq 0$, so that
\[ \not\vdash p \in \sim U(W \uparrow n). \] (3.8)

As $\vdash p \in \mathcal{M}$, by our hypothesis that ($\vdash p$, $\not\vdash p$) is a clash, there is a class $b$ in $\mathcal{H}$ containing a clause
\[ p \leftarrow q_1, \ldots, q_m \]
such that
\[ \vdash q_1, \ldots, \vdash q_m, \not\vdash \overline{p} \in \mathcal{M} \] (3.9)
and
\[ \rho(\vdash q_1), \ldots, \rho(\vdash q_m), \rho(\not\vdash \overline{p}) < \rho(\vdash p). \] (3.10)

Choose a minimal such $b$ in $\mathcal{H}$, which is possible because $\mathcal{H}$ has only a finite number of classes.

In $\mathcal{H}^*$ we have the clause
\[ \sim p \leftarrow \sim q_1, \ldots, \sim q_m, \not\vdash b \overline{p}. \]

As $\sim p \in U(W \uparrow n)$, by (3.8), one of the following conditions arises, by definition of unfounded set:

i. $\sim q_i \in U(W \uparrow n)$ for some $i = 1, \ldots, m$.

ii. $\not\vdash q_i \in W \uparrow n$ for some $i = 1, \ldots, m$.

iii. $\vdash b \overline{p} \in W \uparrow n$.

Let us consider each of the three cases in turn.

(i) In this case, $\not\vdash q_i \in \sim U(W \uparrow n) \subseteq W \uparrow n + 1$, so that, by (3.9), we have a clash ($\vdash q_i$, $\not\vdash q_i$). We also have
\[ \rho(\vdash q_i, \not\vdash q_i) \leq \rho(\not\vdash q_i) \leq n + 1 = \rho(\vdash p, \not\vdash p), \]
so that $\rho(\vdash q_i, \not\vdash q_i) = n + 1$ by the minimality of the clash ($\vdash p$, $\not\vdash p$). On the other hand, we have $\rho(\vdash q_i) < \rho(\vdash p)$, by (3.10). This contradicts the choice of the clash ($\vdash p$, $\not\vdash p$) as the one with least $\rho(\vdash p)$ among those of rank $n + 1$.

(ii) In this case we have a clash ($\vdash q_i$, $\not\vdash q_i$), by (3.9), with rank
\[ \rho(\vdash q_i, \not\vdash q_i) \leq \rho(\not\vdash q_i) \leq n < \rho(\vdash p, \not\vdash p), \]
contradicting the minimality of the rank of the clash ($\vdash p$, $\not\vdash p$).

(iii) In the last case, we assume that $\vdash b \overline{p} \in W \uparrow n$. There is a class $c < b$ and a rule
\[ \overline{p} \leftarrow r_1, \ldots, r_k \]
in \(c\) such that

\[
\vdash r_1, \ldots, r_k \in W \uparrow n - 1.
\] (3.11)

In \(\mathcal{H}^*\) we have the clause

\[
\vdash \bar{p} \leftarrow \vdash r_1, \ldots, r_k, \vDash^c p.
\] (3.12)

As \(\vDash \bar{p} \in \mathcal{M}\), by (3.9), either \(\vDash r_i \in \mathcal{M}\) for some \(i = 1, \ldots, k\) or \(\vdash^c p \in \mathcal{M}\).

In the first case, we have a clash \((\vdash r_i, \vDash r_i)\), by (3.11), with rank

\[
\rho(\vdash r_i, \vDash r_i) \leq \rho(\vdash r_i) \leq n - 1 < \rho(\vdash p, \vDash p),
\]

which is impossible.

In the second case, there is a class \(d < c\) with a clause

\[
p \leftarrow t_1, \ldots, t_l
\]

such that

\[
\vdash t_1, \ldots, t_l \in \mathcal{M}
\]

and

\[
\rho(\vdash t_1), \ldots, \rho(\vdash t_l) < \rho(\vdash^c p).
\] (3.13)

Thus, for every \(i = 1, \ldots, l\), we have

\[
\rho(\vdash t_i) < \rho(\vdash^c p) \quad \{\text{by (3.13)}\}
\]

\[
< \rho(\vDash \bar{p}) \quad \{\text{by (3.12)}\}
\]

\[
< \rho(\vdash p). \quad \{\text{by (3.10)}\}
\]

As \(\vDash \bar{p} \in \mathcal{M}\) and \(d < b\), this contradicts the minimality property assumed for \(b\). This ends the proof.

\[\blacksquare\]

**Theorem 74 (Consistency in \(\mathcal{H}^*\))** For any property \(p\) in a HST \(\mathcal{H}\), \(\vdash p\) and \(\vdash \bar{p}\) are not both true in the well-founded model of the corresponding logic program \(\mathcal{H}^*\).

**Proof:**

If \(\vdash p\) is true in the well-founded model of \(\mathcal{H}^*\) then \(\vDash \bar{p}\) is true, as it belongs to all bodies of clauses for \(\vdash p\). Thus if \(\vdash \bar{p}\) is also true, we have a clash \((\vdash \bar{p}, \vDash \bar{p})\), which is impossible by theorem 73.

\[\blacksquare\]
3.3 Proving the consistency

3.3.2 From well-founded to stable semantics

It is useful to further analyze the translated program $\mathcal{H}^\ast$.

A general program is call-consistent [Dun92] if in it no atom depends oddly on itself, i.e., through an odd number of calls to negative goals. A general program is strict if for no two atoms $g$ and $h$ it happens that $g$ depends on $h$ simultaneously through an odd and an even number of negations. Considering that every $g$ always depends on itself through an even dependency (zero negations) implies that a strict program is also call-consistent.

In $\mathcal{H}^\ast$, as can be seen in definition 72, strong and local atoms have a direct even dependency only on strong atoms and a direct odd dependency only on weak atoms. Weak atoms have even dependency on weak atoms and odd dependency on local and strong atoms. An atom cannot depend on another simultaneously through an even and an odd number of negations. Thus $\mathcal{H}^\ast$ is strict and call-consistent. The main consequences are the following.

Proposition 75 (Strict) If $\mathcal{H}^\ast$ is a strict logic program then

- $\mathcal{H}^\ast$ has (total) stable models, and
- an atom that is undefined in its well-founded model is true in some stable models and false on the others.

Proof:

See [Dun92, Mon92].

The stable models [GL88] of a general logic program $P$ are the fixed-points of an operator $\Gamma_P(I)$. Given an a priori interpretation $I$, the negative literals in rule bodies are simplified according to $I$. If at least one of the negative literals is false in $I$, the rule is deleted from the program. The true negative literals are then removed from the remaining clauses. The simplified program $\frac{P}{I}$ is thus definite and its immediate consequence operator has a least fixed-point $\Gamma_P(I) = T_P \uparrow \omega$. If $T_P$ is able to regenerate the initial interpretation, then $I = \Gamma_P(I)$ is a stable model. Notice that stable models are total two-valued models and that they are supported due to its very definition.

Example 34 Well-founded model with undefined literals.

The HST in figure 3.8 displays a mutual overriding situation, similar in behavior to the logic program with an even negative loop $\{ p \leftarrow \neg r, r \leftarrow \neg p \}$, which is used to
exemplify the alternative stable models. Remember that its well-founded model is empty. However, the extra property $s$ introduces an asymmetry in the HST.

According to definition 72, the part of the translated program regarding target class $a$ is

\[
\begin{align*}
&\vdash p \leftrightarrow \not p \\
&\vdash r \leftrightarrow \not r \\
&\vdash s \leftrightarrow \not r, \not s \\
&\vdash \not p \leftrightarrow \not r, \not p \\
&\vdash \not r \leftrightarrow \not p, \not s, \not r \\
&\vdash \not p \leftrightarrow \not r, \not s, \not p, \not r \\
&\vdash \not r \leftrightarrow \not p, \not s, \not p, \not r \\
&\vdash \not p \leftrightarrow \not r, \not s
\end{align*}
\]

It has the well-founded model of figure 3.9, where the symbol $\not$ means undefined.

<table>
<thead>
<tr>
<th>$\mathcal{M}$</th>
<th>$p$</th>
<th>$\not p$</th>
<th>$r$</th>
<th>$\not r$</th>
<th>$s$</th>
<th>$\not s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vdash$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
</tr>
<tr>
<td>$\vdash^a$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
</tr>
<tr>
<td>$\vdash^b$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
</tr>
<tr>
<td>$\vdash^c$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
<td>$#$</td>
</tr>
</tbody>
</table>

Figure 3.9: Well-founded model with undefined symbols.

The stable models of the translated program, which extend the well-founded model $\mathcal{M}$, are $M_1$ to $M_3$, presented in figure 3.10. Stable models are total. The undefined symbols in $\mathcal{M}$ may become either true or false in $M_i$. Although the stable models are two-valued total interpretations, we go on using the same notation as we did with the partial well-founded models and write $\not x \in M$ instead of $x \notin M$.

The stable models [GL90] are defined through the fixed-point of an operator so that their conclusions are self-supported. However, $M_1$ extends $\mathcal{M}$ in a way that does not agree with theorem 73 and has $\vdash p$ true, while $\vdash p$ is false. Both $M_2$ and $M_3$ do not suffer from this inconvenience but $M_2$ has more strong conclusions than $M_3$. Of the three, $M_3$ is the closest to the well-founded model, with respect to the strong conclusions. Its relationship with $\mathcal{M}$ is the following. All the strong and local atoms that are undefined in $\mathcal{M}$ become false in $M_3$ and the weak atoms become true.
<table>
<thead>
<tr>
<th>$M_1$</th>
<th>$p$</th>
<th>$\neg p$</th>
<th>$r$</th>
<th>$\neg r$</th>
<th>$s$</th>
<th>$\neg s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vdash$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td></td>
</tr>
<tr>
<td>$\models$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
</tr>
<tr>
<td>$\vdash^b$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdash^a$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$M_2$</th>
<th>$p$</th>
<th>$\neg p$</th>
<th>$r$</th>
<th>$\neg r$</th>
<th>$s$</th>
<th>$\neg s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vdash$</td>
<td>$\sim$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\sim$</td>
<td></td>
</tr>
<tr>
<td>$\models$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
</tr>
<tr>
<td>$\vdash^b$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\checkmark$</td>
<td>$\sim$</td>
<td></td>
</tr>
<tr>
<td>$\vdash^a$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$M_3$</th>
<th>$p$</th>
<th>$\neg p$</th>
<th>$r$</th>
<th>$\neg r$</th>
<th>$s$</th>
<th>$\neg s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vdash$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
</tr>
<tr>
<td>$\models$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
</tr>
<tr>
<td>$\vdash^b$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
</tr>
<tr>
<td>$\vdash^a$</td>
<td>$\sim$</td>
<td>$\sim$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.10: Stable extensions of the well-founded model.

The above mentioned models are also extended, or three-valued, stable models. There is another extended stable model, which is partial, but, due to the strictness of the translated program, it lies somewhere between $\mathcal{M}$ and one of the total stable models.

The atoms in $\mathcal{H}^{*}$ represent assertions about the derivability of the respective property relative to one of three derivation relations. So they are best understood as being two-valued. This explains the importance of the existence of total stable models for $\mathcal{H}^{*}$. It does not mean that we are weakening our partial or three-valued view of the properties themselves, because their valuation results from the composition of two atoms, $\vdash p$ and $\vdash \bar{p}$.

**Example 35** *Overriding based on the weak derivation.*

The strictness of the translated program would be lost if, in inference rule [LO], instead of a strong derivation of the body, a weak derivation were required.

\[
\frac{\vdash_a Q \quad \{ p \leftarrow Q \in \mathcal{D}_c. \}}{\vdash_b p \quad \{ a \leq \ c < b \}} \tag{LO'}
\]

We reuse the HST of example 32 because it is the smallest one with an odd negative loop.
From the translation it is easy to see that \( \sim \neg p \) depends negatively on itself (through \( \vdash^b p \)). Odd negative loops are undefined in the well-founded model. Moreover, such a program is not strict and has no total supported models. It is not possible to say whether \( \neg p \) is weakly derivable or not, a situation that seems anomalous due to the intuitive two-valued character of the derivation relations. So, the inference rule [LO'] proves to be less adequate than the previously proposed [LO].

\[ \square \]

### 3.3.3 In HST

To carry these results to the HST itself we need to establish a correspondence between the models of \( \mathcal{H}^* \) and the original \( \mathcal{H} \) plus the inference rules.

**Definition 76 (Correspondence)** Given a HST \( \mathcal{H} \) and a target class a the triple \( I = (\vdash_a, \sim_a, (\vdash^z_a)_{a \leq z}) \) of derivation relations corresponds to the total interpretation \( M \) of \( \mathcal{H}^* \) with respect to a if and only if

\[
\begin{align*}
\alpha \ p \in M & \iff p \in \alpha \\
\sim \alpha \ p \in M & \iff p \notin \alpha
\end{align*}
\]

(3.14)

where \( \alpha \) is one of \( \vdash_a, \sim_a, \vdash^z_a \).

**Proposition 77 (From LP to HST)** Let \( \mathcal{H} \) be a HST and \( a \) a target class. \( M \) is one of the stable models of \( \mathcal{H}^* \) with respect to \( a \) if and only if its corresponding triple \( I \) of derivation relations is induced by the inference rules.

**Proof:**

First we show that if \( M \) is a stable model of \( \mathcal{H}^* \) then the triple \( I \), obtained from \( M \) by (3.14), fulfills the conditions of definition 70, i.e., it satisfies the inference rules and \( p \in \alpha \) only if there is a successful derivation of \( \alpha \ p \).

A. Let us assume that \( I \) does not satisfy the inference rules. Then there is a rule

\[
p \leftrightarrow q_1, \ldots, q_n \in D_c
\]

(3.15)
such that one of the following conditions holds:

\begin{align*}
q_1, \ldots, q_n &\not\vdash \varphi \not\vdash \varphi, p \not\vdash \varphi \quad \text{or} \\
q_1, \ldots, q_n &\not\vdash \varphi, \varphi \not\vdash c, p \not\vdash \varphi \quad \text{or} \\
q_1, \ldots, q_n &\not\vdash \varphi, \exists c > c, p \not\vdash \varphi.
\end{align*}

Notice that, as \( M \) is two-valued, \( p \not\in \infty \) is equivalent to \( \sim \infty p \in M \). From (3.14),

\begin{align*}
\vdash q_1, \ldots, q_n &\in M, \sim \vdash \varphi \in M, \sim \vdash \varphi \in M \quad \text{or} \\
\vdash q_1, \ldots, q_n &\in M, \sim \vdash c \varphi \in M, \sim \vdash c \varphi \in M \quad \text{or} \\
\vdash q_1, \ldots, q_n &\in M, \exists \alpha > c, \sim \vdash \varphi \in M.
\end{align*}

So, at least one of the clauses in \( \mathcal{H}^* \) obtained from (3.15) through the translation of definition 72 is not satisfied, contradicting the hypothesis that \( M \) is a model.

B. Suppose that there is a \( p \in \vdash \) such that there is no successful derivation of \( \vdash p \). Then, by [SR], for every rule \( p \leftarrow q_1, \ldots, q_n \in D_c \)

\[ q_1 \not\vdash \vee \ldots \vee q_n \not\vdash \vee \varphi \in \sim. \]

In \( \mathcal{H}^* \), the translated clause

\[ \vdash p \leftarrow q_1, \ldots, q_n, \sim \vdash \varphi \]

(3.16)
cannot be used to conclude \( \vdash p \) because

a) if \( \varphi \in \sim \), by (3.14), \( \sim \varphi \in M \) and so the clause (3.16) does not belong to \( \mathcal{N}_M^* \); or

b) there is no successful derivation of \( \vdash q_i \), reproducing the situation of \( \vdash p \). In this case either the reasoning may recur, until a descendant of \( q_i \) is found that has no rules, or the derivation is infinite. In both cases, \( \vdash q_i \) cannot belong to \( \Gamma_{\mathcal{N}_M^*}(M) = T_{\mathcal{N}_M^*} \uparrow \omega \) and thus neither can \( \vdash p \), using clause (3.16).

As \( \vdash p \in M \), by (3.14), then \( M \) is not a stable model, contradicting the hypothesis. Similar arguments can be used for the cases \( p \in \sim \) and \( p \in \vdash b \).

So, if \( M \) is a stable model, \( I \) is induced by the inference rules.

We will now prove the only if part of the theorem, again by contradiction. Suppose that \( M \) is not a stable model. Then \( M \) contains either more or less literals than \( \Gamma_{\mathcal{N}_M^*}(M) \).

A. If \( \alpha p \in \Gamma_{\mathcal{N}_M^*}(M) \) but \( \sim \alpha p \notin M \) then there is a clause for \( \alpha p \) whose body agrees with \( M \) on its negative literals (otherwise the clause would have been
removed from \( \frac{H^*}{M} \), while the positive literals belong to \( T_k^{k-1}(M) \), where \( k \) is the order of the iteration at which \( \alpha p \) is generated. Let us assume, without loss of generality, that \( k \) is the least order at which a literal not in \( M \) is generated.

Let \( p \leftarrow q_1, \ldots, q_n \) be the rule in \( H \) that, by the translation of definition 72, originates the clause for \( \alpha p \). Then \( I \), by (3.14), does not satisfy the reduction inference rule [SR], [WR] or [LO], depending on \( \alpha \), because it contains all the goals in its antecedent and does not contain the consequent \( \alpha p \).

**B. If \( \alpha p \in M \) and \( \alpha p \notin \Gamma_{H^*}(M) \) then for every clause**

\[
\alpha p \leftarrow \alpha q_1, \ldots, \alpha q_n, \sim \sim \tilde{p}
\]  (3.17)

in \( H^* \), either

a) \( \alpha \tilde{p} \in M \) and thus the clause is deleted from \( \frac{H^*}{M} \) (\( \sim \alpha \tilde{p} \) is \( \sim \mid \tilde{p} \), \( \sim \mid \sim \tilde{p} \) or nothing, when \( \alpha p \) is \( \mid p \), \( \mid \sim p \) or \( \mid \sim \tilde{p} \)); or

b) at least one \( \alpha q_i \) does not belong to \( T_{H^*} \uparrow \omega \).

For case a), we have that \( \tilde{p} \notin \alpha \) in \( I \), by (3.14). So, the rule in \( H \) that corresponds to (3.17) cannot be used in a successful derivation of \( \alpha p \).

For case b), we notice that \( \alpha q_i \) is in the same situation as \( \alpha p \) in (3.17). We have already shown that, if the reason for the absence of \( \alpha q_i \) in \( T_{H^*} \uparrow \omega \) is the falsity of a negative goal, then the corresponding rule cannot enable a successful derivation of \( \alpha p \). The other alternatives are a lack of applicable clauses for \( \alpha q_i \), in which case no rules exist for a derivation of \( \alpha q_i \), or \( \alpha q_i \) belongs to an unfounded set and so it does not belong to the least fixed-point \( T_{H^*} \uparrow \omega \). This case gives rise to an infinite, and thus failed, derivation of \( \alpha q_i \) and of \( \alpha p \). By (3.14), \( p \in \alpha \) in \( I \), but there is no successful derivation of \( \alpha p \).

Therefore, \( I \) is not induced by the inference rules, contradicting the hypothesis.

In conclusion, stable models of \( H^* \) correspond to triples of derivation relations induced by the inference rules with respect to \( H \).

The digression through the translated program \( H^* \) means that the interpretation that we are assigning to the negative premises in the inference rules agrees with the well-founded semantics. From proposition 75, the well-founded model of a strict program is the intersection of its stable models. It constitutes the core of the triples of derivation relations corresponding to those models.

A set of structural rules like the one of figure 3.1 normally concerns one relation symbol and defines the least relation which satisfies them [Plo81]. However, in our
case, the rules simultaneously refer to three relation symbols connected by negative premises. When the well-founded model of $\mathcal{H}^*$ is not total, to the stable models correspond alternative triples of relations. To make the inference rules meaningful, we need an order ranking these triples such that a least one exists. In order to get the most skeptical solution, the designated triple should be as close to the well-founded model as possible, from the perspective of the strong relation.

**Definition 78 (Order)** A triple $I = \langle \vdash_a, \vdash_a', (\vdash^x_a)_{a \leq x} \rangle$ is less than or equal to a triple $I' = \langle \vdash_a', \vdash_a', (\vdash^x_a')_{a \leq x} \rangle$, written $I \sqsubseteq I'$, if and only if $\vdash_a \subseteq \vdash_a'$.

The relation $\sqsubseteq$, if understood on the set of all triples, is just a pre-order. However, when restricted to the set of triples induced by the inference rules, $\sqsubseteq$ is a partial order, because then the strong relation determines the other two. After fixing $\vdash$, the family of local relations is obtained by a single application of the immediate consequence operator. It settles all the negative goals that may be needed in derivations of weak goals. The latter coincide with the least fixed-point of the definite program obtained by the union of all the relevant theories in the HST, after deleting in each class $c$ the rules for $p$ such that $\overline{p} \in \vdash^c$ i.e., the overridden rules.

**Proposition 79 (Least triple)** There is a $\sqsubseteq$-least triple induced by the inference rules of figure 3.1.

**Proof:**

The analysis of the rules in figure 3.1, using some object-level rule $p \leftarrow Q \in D_b$, reveals the following dependencies, where $X \leftarrow Y$ means $X$ may depend on $Y$:

- **[SR]**
  - $\vdash p \leftarrow \not\vdash \overline{p}$
  - $\not\vdash p \leftarrow \vdash \overline{p}$
  - $\not\vdash p \leftarrow \not\vdash Q$
  - $\vdash p \leftarrow \not\vdash Q$

- **[WR]**
  - $\vdash p \leftarrow \not\vdash \overline{b} \overline{p}$
  - $\not\vdash p \leftarrow \not\vdash \overline{b} \overline{p}$
  - $\not\vdash p \leftarrow \vdash \overline{Q}$
  - $\vdash p \leftarrow \not\vdash Q$

- **[LO]**
  - $\not\vdash \overline{b} \overline{p} \leftarrow \vdash \overline{Q}$
  - $\not\vdash \overline{b} \overline{p} \leftarrow \not\vdash Q$.

Using transitivity we may get some derived dependencies, like $\vdash p \leftarrow \not\vdash \overline{p} \leftarrow \not\vdash \overline{s} \leftarrow \vdash r$, under appropriate conditions, including

- $\vdash p \leftarrow \vdash r$
- $\vdash p \leftarrow \not\vdash r$
- $\vdash p \leftarrow \vdash r$
- $\vdash p \leftarrow \not\vdash r$. 


Dependencies for the same relation preserve the sign (positive for derivation, negative for non-derivation) and dependencies for different relations switch the sign. We are considering, in this context, the local relation as a special case of the strong relation. The conclusion is that we will never get \( \vdash p \not\iff \mathcal{R} \), the falsity of a strong property making another strong property become true. That situation would occur for some properties \( p, \mathcal{R} \) if two alternative minimal strong relations were to exist. As this can never happen then it is possible to simultaneously purge all the strong properties that do not belong to at least one of the triples induced by the inference rules. The triple determined by such strong derivation relation is \( \sqsubseteq \)-less than every other and so it is the \( \sqsubseteq \)-least triple.

So, we are finally able to assign a unique meaning to the inference rules.

**Definition 80 (Operational semantics)** The operational semantics associated to a HST, with respect to a target class \( \alpha \), is the \( \sqsubseteq \)-least triple induced by the inference rules of figure 3.1.

This triple is the back-translation, by (3.14), of the total stable model of \( \mathcal{H}^* \) obtained from its well-founded model by setting the undefined strong and local atoms to false and the undefined weak atoms to true. If some atoms are undefined that is because they belong to (even) negative loops. The weak atoms depend only on the local atoms. If these are set to false, the former must be true.

The operational semantics defined corresponds to slightly modifying the well-founded semantics by giving priority to the falsification of strong atoms. The very notion of proof is affected by this result in what concerns the way negative goals are dealt with. A negative goal succeeds if the derivation of the corresponding positive goal fails. When a loop involves both weak and strong (or local) subderivations, which do not fail by other reasons, the latter are the ones to fail and the weak subderivations succeed. Notice that derivations with a negative loop are cyclic, i.e., they are unfounded in the sense that the marking of a node depends on itself. An extra criterion is needed to choose which derivations fail and which ones are successful.

Referring to the example 32, \( I_2 \sqsubseteq I_1 \), and so \( I_2 \) is the triple defined by the operational semantics. In example 34, the triple corresponding to \( M_3 \) is the set of conclusions inferred from the HST.

**Theorem 81 (Strong implies weak)** For any property \( p \) in \( \alpha \) HST \( \mathcal{H} \), if the derivation of \( \vdash p \) succeeds then the derivation of \( \neg p \) succeeds, too.

**Proof:**
The well-founded model $\mathcal{M}$ of $\mathcal{H}^*$ is free of clashes, by theorem 73. As the strong derivation relation in the triple defined by the operational semantics, by proposition 79, has no more strong conclusions than those in $\mathcal{M}$, it does not contain clashes, too.

The $\sqsubseteq$-least triple induced by the inference rules does not contain $\vdash \neg p$ if it contains $\vdash p$.

**Corollary 82 (Consistency of $\mathcal{H}$)** The operational semantics of a HST $\mathcal{H}$ is consistent.

**Proof:**

The well-founded model of the translated program $\mathcal{H}^*$ does not contain strong atoms corresponding to complementary properties, by theorem 74. As the triple defined by the operational semantics does not contain any new strong conclusions, it is consistent.

The two last theorems are interesting not just from the theoretical viewpoint but also from a practical perspective. The result on consistency relieves the inference procedure from testing the complementary $\vdash \neg p$ once the derivation of $\vdash p$ is successful. If $\vdash p$ fails, however, then $\vdash \neg p$ must be tried out. If it succeeds the external answer is *false*, otherwise it is *undefined*.

The inner works of the inference procedure do not require a (strong) truth value for a property. The notion of derivability of certain relations is enough. Nevertheless, to strongly derive a property $p$ the opposite weak derivation of $\neg p$ must fail. The latter may depend on the local derivation of $p$ itself (try $\vdash \text{octopus: swim}$ with respect to the HST in figure 3.6 for an instance of this situation). So it is likely that sometimes a derivation includes repeated subderivations. An implementation is able to avoid redundant work if the ongoing conclusions are kept as lemmas which can be reused in later stages. In this context, theorem 81 becomes relevant. Whenever $\vdash p$ is derived $\neg p$ may also be stored as a lemma, as well as $\vdash \neg b p$, for the appropriate classes $b$. Conversely, if we get $\not\vdash p$ we can assert $\not\vdash p$ too.

A third note related to implementation restricts the search space for weak derivations.

**Proposition 83 (Guiding weak derivations)** During a strong derivation of $p$ that starts with a rule from a class $b$, the inherent check for the weak opposite $\neg p$ may start with a rule from a class $c$ not above $b$.

**Proof:**
To achieve a successful derivation of \( p \), a rule \( p \leftarrow Q \in D_b \) is selected. Suppose that, in the attempt to weakly contradict it, a rule \( \bar{p} \leftarrow Q' \) is picked up from a superclass \( c \) of \( b \). If \( \vdash Q \) succeeds then, by inference rule [LO], one concludes \( \vdash^c p \) and, by [WR], \( \not\vdash \bar{p} \), ending with \( \vdash p \), via [SR].

This restriction was not included in the inference rules to give greater generality to and simplify the form of the definitions for the weak relation.

### 3.4 Correctness of the inference system

We have presented two independent methods to extract the information encoded in a HST. The characterization \(<a>\) of a class \( a \) is the least fixed-point of a bottom-up operator, which contains the properties that can safely be attributed to the class. The strong inference relation \( \vdash_a \) gives the properties that are derived by a top-down procedure focused in the class \( a \). It is expected that the two sets are closely related. We show it in this section.

#### 3.4.1 Rank of a derivation

In the proofs that follow, we will often omit the references to the rules for conjunction [SC] and [WC], which are trivially verified.

**Definition 84 (Rank of a derivation)** The rank of a strong or local derivation of a property \( p \) using a rule \( p \leftarrow Q \) is recursively defined to be

\[
\begin{align*}
\text{rank}(\vdash p) & = 1 + \max(\{\text{rank}(\vdash q_i), \text{rank}(\not\vdash \bar{p})\}) \\
\text{rank}(\vdash^e p) & = 1 + \max(\{\text{rank}(\vdash q_i)\})
\end{align*}
\]

where the \( q_i \) are the positive subgoals introduced by the application of the respective inference rule, \( \max(\emptyset) = 0 \) and

\[
\text{rank}(\not\vdash p) = \max(\{\text{rank}(\vdash^e r_i)\}),
\]

where the \( r_i \) are the properties with a successful local derivation that contribute directly to the failure of an alternative for the weak derivation of \( p \).

According to [SR], the properties \( p \) depends on directly are those in the body of the rule used to derive it and the locally derived properties that force the failure of the weak derivation of \( \bar{p} \). Notice that the properties mentioned in all the main trees of the (failed) alternatives for the weak subderivation are not counted. But the locally derived properties sharing the responsibility for such failure are, even if they are introduced only after some weak steps.
Figure 3.12: Collective overriding.
<table>
<thead>
<tr>
<th>deriv.</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\vdash \neg q$</td>
<td>0</td>
</tr>
<tr>
<td>$\vdash q$</td>
<td>1</td>
</tr>
<tr>
<td>$\vdash^e \neg p$</td>
<td>2</td>
</tr>
<tr>
<td>$\vdash^e \neg u$</td>
<td>1</td>
</tr>
<tr>
<td>$\vdash \neg p$</td>
<td>2</td>
</tr>
<tr>
<td>$\vdash p$</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 3.13: Ranks of the derivations.

**Example 36 Dependencies.**

At first sight we can say that class $a$ in the HST of figure 3.12 enjoys $p$ if it is locally derived in $b, d$ and $f$. Then the three rules would collectively override the rules for $\neg p$ above them. Actually, $p$ is not locally derived in $d$ but it is still a strong conclusion, because the rules in $e$ do not derive (weakly) $\neg p$. The first depends on an overridden property $u$ and the second on a property for which there are no rules.

The strong derivation of $p$ using the rule $p \leftarrow$ in $f$ depends on the properties in its body, i.e., none, and on a sufficient subset of the following properties: those in the bodies of the other rules for $p$ and the complements of the properties in the bodies of the rules for $\neg p$. Among the properties $\{x, q, \neg r, \neg s, \neg u, \neg v\}$, knowing that $q$ and $\neg u$ hold for $a$ is enough to conclude $p$. The first produces $p$ in $b$ thus overriding $\neg p$ from $c$. The second overrides the support for $\neg p$ in $e$.

The analysis of the strong derivation in figure 3.12 shows that it includes just two successful local subderivations, one of $\neg u$ and the other of $p$ via $q$. As the derivation has a single root, corresponding to the rule in $f$, the team work developed by the other rules for $p$ becomes manifest through the local subderivations of $p$ itself.

Connecting the local subderivations to the main derivation there is a set of three attempted weak derivations, corresponding to the three rules for $\neg p$, all of which fail. These steps are not counted in the rank of a derivation precisely because they fail and so are not useful as a measure of its depth. Only the steps in the even levels of the derivation are constructive. The ranks of the subderivations in figure 3.12 are listed in figure 3.13.

**Lemma 85 (Local and strong)** If a successful local subderivation of $r$ is contained in a derivation then a successful strong derivation of $r$ exists.

**Proof:**
Due to the structure of derivations, a successful local derivation of \( r \) exists only in the context of a derivation of \( \vdash \neg r \) (see [WR]). As \( \vdash \neg r \) succeeds then \( \vdash \neg r \) fails. If using another rule for \( \neg r \) would make \( \vdash \neg r \) succeed then that derivation would have been chosen and \( \vdash \neg r \) would not be present.

If \( \vdash \neg r \) succeeds, by [LO] there is a rule \( p \leftarrow Q \) such that \( \vdash Q \). As \( \not\vdash \neg r \), by [SR], \( \vdash r \).

There may be independent local derivations of \( p \) without a successful derivation of \( \vdash p \) but only if, simultaneously, \( \vdash \neg p \) succeeds.

In a strong derivation of \( p \) it is possible that \( p \) may reappear but only as a local subderivation, two levels below \( \vdash p \). If \( \vdash p \) is found twice, the subderivation containing the second instance must fail because either

a. one of the other subgoals conjuncted with \( \vdash p \) fails and in that case \( \vdash p \) may succeed (see figure 3.14), or

b. \( \vdash p \) belongs to an unfounded set, the derivation is infinite and thus \( \vdash p \) fails.

As the number of properties is finite, so is the rank of a successful derivation. Infinite branches may show up in the weak part because it is failed anyway (see figure 3.15).

### 3.4.2 Completeness

**Theorem 86 (Strong completeness)** The properties that belong to the characterization have a successful strong derivation, \( <a> \subseteq \vdash a \).

**Proof:**

From (2.17), the characterization is the least fixed-point of the safe conclusions operator. The proof is by induction on the number \( n \) of iterations of \( S_D \),

\[
J^{n+1} = S_D(J^n) = J^n \cup \Delta(D/J^n).
\]

We will prove simultaneously that, for every step \( n \),

\[
J^n \subseteq \vdash \quad \text{ (3.19)}
\]

\[
\overline{J^n} \subseteq \not\vdash \quad \text{ (3.20)}
\]

For \( n = 0 \) we have \( J^0 = \emptyset \) and also \( \emptyset \subseteq \vdash \) and \( \emptyset \subseteq \not\vdash \).
For the inductive step, we assume that \( p \in J^{n+1} \). Only the case \( p \not\in J^n \) is of interest. So \( p \in \Delta(D/J^n) \) and, by (2.14),

\[
p \in T^*((D/J^n)[T^*(D/J^n)]).
\]  

(3.21)

From (2.13) we have that

\[
p \in T^*(D/J^n) \\
\overline{p} \not\in T^*(D/J^n)
\]  

(3.22)

which imply, by (2.11), that there is a rule

\[
p \leftarrow Q' \in (D/J^n)_b, \ Q' \subseteq T^*(D/J^n), \ \overline{p} \not\in \{(D/J^n)_c, \ c < b \}
\]  

(3.23)

and that for every rule

\[
\overline{p} \leftarrow R' \in (D/J^n)_d \Rightarrow [R' \not\subseteq T^*(D/J^n) \vee p \leftarrow \{(D/J^n)_e, \ e < d\}]
\]  

(3.24)
3.4 Correctness of the inference system

We use the following notation. Rules \( p \leftarrow Q \) are written \( p \leftarrow Q', Q'' \) by splitting their bodies according to a set of properties \( J \) such that \( Q'' \subseteq J \) and \( Q' \cap J = \emptyset \). From definition (2.15) of division, if \( p \leftarrow Q' \in (D/J)_b \) then there is a rule \( p \leftarrow Q', Q'' \in D_b \). Conversely, if \( p \leftarrow Q', Q'' \in D_b \) and there is no rule \( p \leftarrow Q' \in (D/J)_b \) then \( Q' \cap J \neq \emptyset \).

If there is a fact \( p \leftarrow (D/J^n)_e \) there is a rule \( p \leftarrow S'' \in D_e \) with \( S'' \subseteq J^n \). By the hypothesis (3.19), \( \vdash S'' \) and, by the inference rule [LO], \( \vdash^d p \). If the only rules for \( p \) are in a class \( d \) above \( e \) then, from [WR], \( \not\vdash \overline{p} \).

The other alternative in (3.24) is the existence of a \( r \in R' \) such that

\[
r \notin T^*(D/J^n).
\]  

(3.25)

Statement (3.25) is similar to the second line of (3.22) and thus implies a suitable translation of (3.24). The reasoning recurs until a property \( x \) for which there are no rules, or they are all overridden, is found. Notice that each property involved in the recursion appears only once because possible repetitions are due to unfounded rules. As the number of properties is finite, so is the recursion.

If there are no rules for \( x \) in \( D/J^n \) then \( \not\vdash x \) because the other possible rules for \( x \) in \( D \), which are eliminated by the division, have at least one property in \( J^n \) and we are assuming (3.20). As seen above, rules that are overridden do not allow weak derivation either. Property \( x \) being not weakly derived implies, by [WR], that the rules which have \( x \) in the body cannot be used in a weak derivation, too. So, \( \not\vdash r, \not\vdash R' \) and \( \not\vdash \overline{p} \), confirming (3.20).

From (3.23), there is a rule \( p \leftarrow Q', Q'' \) in \( D_b \) and \( Q'' \subseteq J^n \). By the hypothesis (3.19), \( \vdash Q'' \). If \( Q' = \emptyset \) and as \( \not\vdash \overline{p} \) then, by [SR], \( \vdash p \), proving (3.19). If \( Q' \neq \emptyset \) then the analysis of (3.21) and the supportedness of the operator \( T^* \) insure that every \( q \in Q' \) is in the same situation as \( p \) in (3.22). So there is a recursion in the bodies of rules until the facts that support \( p \) and, again by [SR], we have \( \vdash Q' \) and \( \vdash p \).

As, by definition, \( \langle a \rangle \) is the least fixed-point of \( S_D \), for a sufficiently high \( n \),

\[
\langle a \rangle = J^n \subseteq \langle a \rangle.
\]

(3.26)

Theorem 87 (Weak soundness) The properties with a successful weak derivation belong to the weak characterization, \( \vdash_a \subseteq \langle \overline{a} \rangle \).

Proof:

Let us assume that a weakly derived property \( p \) does not belong to the weak characterization. From lemma 65 this means

\[
p \notin T^*(D[\langle a \rangle]).
\]  

(3.26)
By (2.11), for every rule \( p \leftarrow Q \in (D[<a>]_c) \), either
\[
Q \not\subseteq T^*(D[<a>]) \quad (3.27)
\]
or
\[
\overline{p} \leftarrow \in (D[<a>]), c < b. \quad (3.28)
\]
If a rule is inhibited from producing \( p \) by case (3.28) then there is a fact \( \overline{p} \leftarrow \) in \( D_c, c < b \) and, by inference rule [LO], we have \( \vdash^c \overline{p} \). This implies that \( \not\vdash p \), by [WR], contradicting the assumption.

In case (3.27) there is a \( q \in Q \) in the same condition as \( p \) in (3.26). So the reasoning recurs until a property \( x \) is found in one of the following situations.

- Every existing rule for \( x \) is in case (3.28). We have shown that then none can be used to weakly derive \( p \). If there are no rules for \( x \) then obviously \( \not\vdash x \).

- If \( x \) is encountered twice in the recursion then the corresponding weak derivation fails because it is infinite.

Working in the reverse direction \( \not\vdash x \Rightarrow \not\vdash q \Rightarrow \not\vdash Q \Rightarrow \not\vdash p \).

Using rules in \( D \) not in \( D[<a>] \) to get \( \vdash p \) is not possible because, by (2.13), a rule \( p \leftarrow Q \) is eliminated by the restriction only if \( \overline{p} \in <a> \). But then, by theorem 86, \( \vdash \overline{p} \) and, by [SR], \( \not\vdash p \).

\[\]

### 3.4.3 Soundness

**Theorem 88 (Strong soundness, weak completeness)** Every strongly derived property belongs to the characterization and every property in the weak characterization is weakly derived,
\[
\vdash_a \subseteq <a>
\]
\[
<a> \subseteq \vdash_a.
\]

**Proof:**

The proof is by induction on the rank of derivations. What we need to prove is that for all \( k \), the existence of a successful derivation of \( G \) of rank \( k \) implies that if \( G = \vdash_a p \) then \( p \in <a> \) and if \( G = \not\vdash_a p \) then \( p \notin <a> \). If \( G = \vdash_a p \) then all the properties in the body of the respective rule have a strong derivation of rank less than \( k \). So the local derivations are a direct byproduct of the strong derivations.

The base case is for \( \text{rank}(G) = 0 \). Then either \( G = \vdash \emptyset \) and obviously \( \emptyset \subseteq <a> \), or \( G = \not\vdash p \). From definition 84 the success of a \( G \) of rank 0 does not depend on an overriding done by a local derivation. So \( G \) may succeed if, for each alternative derivation of \( \vdash p \), either there is
3.4 Correctness of the inference system

a) lack of rules for $p$ or, recursively, for at least one of the properties in the body of each of the rules for $p$, or

b) a positive loop in the theories and the weak main tree is infinite; then $p$ depends on an unfounded set.

Under such condition, by (2.11), $p$ cannot belong to a least fixed-point like $T^*(D)$. As $<\bar{a}> = T^*(D) <a>) \subseteq T^*(D)$, by lemma 65 and (2.13), then $p \notin <\bar{a}>$. 

**Inductive step.** If there is a strong derivation of $p$ with $rank(\vdash p) = k + 1$ using a rule

$$ p \leftarrow Q \in D_b $$

then, by [SR] and definition 84, $rank(\vdash Q), rank(\vdash \bar{p}) \leq k$. 

By the inductive hypothesis, $\bar{p} \notin <\bar{a}>$. Then, from corollary 55, there is a minimal order $n'$ in the bottom-up iteration of $S_D$ such that $\bar{p} \notin J^{n'} \cup T^*(D/J^{n'})$.

Again by the inductive hypothesis, $Q \subseteq <a>$. By lemma 61, for each $q_i \in Q$ there is an order in the iteration of $S_D$ after which $q_i$ is no longer generated. Let us call $n''$ to the maximum of such orders. Then, by (2.14) and (2.13),

$$ q_i \in J^{n''}, \bar{q}_i \notin T^*(D/J^{n''-1}). $$

Let $n = \max(n', n'' - 1)$. Rule (3.29) becomes $p \leftarrow Q' \in (D/J^n)_b$ and, using (3.30), $Q' \subseteq T^*(D/J^n)$. By (2.11), as $\bar{p} \notin T^*(D/J^n)$ in particular there is no fact $\bar{p}$ in a class $c$ below $b$ or, if it exists, a fact for $p$ belongs to $(D/J^n)_d, d < c$. In any case $p \in T^*(D/J^n), p \in \Delta_D/J^n$ and thus $p \in J^{n+1} \subseteq <a>$.

We prove now the second statement. If $rank(\vdash p) = k + 1$ then, by [WR], for all rules $p \leftarrow Q \in D_b$

$$ \vdash Q \lor \vdash \bar{p}. $$

Let us suppose that $\vdash Q$. Then, from definition 84, $rank(\vdash \bar{p}) \leq k + 1$ and so (see [LO]) there is a rule $\bar{p} \leftarrow R \in D_c, c < b$ such that $\vdash R$. Every $r \in R$ has a rank($\vdash r$) $\leq k$. So, by hypothesis, $r \in <a>$. By lemma 61, there is an order $n'$ in the bottom-up iteration of (2.17) after which $r$ is no longer generated because it belongs to $J^{n'}$. Thus the rule $\bar{p} \leftarrow R \in D_c$ becomes the fact $p \leftarrow (D/J^{n'})_c$, which overrides the rule for $p$ in $b$ inhibiting it, according to (2.11), from belonging to $T^*(D/J^{n'})$. Notice that $\bar{p} \in T^*(D/J^{n'})$ and so $p \notin J^{n'}$. Thus

$$ p \notin T^*(D/J^{n'}) \cup J^{n'}. $$

We now show that this condition also holds in the case $\vdash \bar{p}$. There is a $q \in Q$ such that $rank(\vdash q) \leq k + 1$. But this means that $q$ is in the same situation as $p$ and so the argument recurs in the case of overriding, for some order $n''$ of the bottom-up iteration. If there is no overriding then $rank(\vdash p) = 0$. As seen
before this implies \( p \not\in T^*(D) \) and, by lemma 54, \( p \not\in T^*(D/J^n^n) \cup J^n^n \), too. If \( q \not\in T^*(D/J^n^n) \cup J^n^n \) then \( Q \) is not contained in it and we have again (3.32).

If both cases of (3.31) occur, we choose \( n \) to be the minimal of the two orders, so that the corresponding iteration is the first in which \( p \) is not generated.

From (3.32) and by corollary 55, we have that \( p \not\in \vec{\langle a \rangle} \).

It is interesting to notice that the soundness of the strong derivation procedure depends on the completeness of the weak derivation, i.e., if some properties in the weak characterization do not have a successful weak derivation, unsoundness is introduced in the strong derivation. As the weak derivation itself depends on the strong derivation, it configures a sort of positive feedback where the unsound strong properties may block further weak properties at the cost of completeness and of possibly becoming self-supported. See in figure 3.3 that if \( I_1 \) is the chosen alternative (only possible if the order \( \sqsubseteq \) is ignored) then forgetting \( \vdash \neg p \) opens the door to \( \vdash \neg \neg p \) and \( \vdash p \).

Due to the negative dependency between the strong and the weak derivations, also the strong completeness depends on a restricted form of the weak soundness. Usually, the completeness is harder to prove because the operational semantics is above all required to be sound, even if at the cost of losing some conclusions. A strong derivation relation induced by the rules of figure 3.1 is always complete but it may surpass the characterization. To guarantee its soundness it is necessary to minimize it according to the order \( \sqsubseteq \). That’s why the latter proof results more complex.

**Theorem 89 (Equivalence)** The operational semantics of definition 80 is sound and complete with respect to the bottom-up fixed-point semantics of definition 49,

\[
\langle a \rangle = \vdash_a \\
\vec{\langle a \rangle} = \vdash_{\neg a}.
\]

**Proof:**

This is a rephrasing of theorems 86, 87 and 88.

### 3.5 Reducing to finite failure

When a loop shows up in the derivation of a certain goal, the procedure gets trapped in an infinite computation unless it has a loop detection mechanism built-in. From the viewpoint of the answers provided by the inference procedure, explicitly recognizing the failure of all the alternative paths and getting lost forever along one
of them may both be assimilated to the same result of inability to derive the goal, at least along that path.

However, if the derivation of certain goals depends on others being nonderivable, as happens with the inference procedure of figure 3.1, that strategy no longer works. If the derivation of a negative subgoal is infinite, the whole derivation becomes endless and may be erroneously interpreted as meaning nonderivability of the main goal. That is rather the situation of the negative subgoal, which should enable a positive answer for the main goal. The implementation of inference procedures appealing to nonderivability require a loop detection mechanism to ensure the termination of the subderivations and the production of correct answers. The well-founded model [VGRS90] is the semantic counterpart of this view. It goes around the loop problem due to the bottom-up nature of its fixed-point construction.

The most widely used implementations of inference procedures for Logic Programming, namely Prolog, are based on a distinct approach to dealing with negative goals. Instead of meaning nonderivability, negation means finite failure to derive. In a logic program, like the translated $\mathcal{H}^*$, a negative subgoal $\neg g$ succeeds if all the alternative paths for the derivation of $g$ explicitly fail in a finite number of steps. One of the most well-known semantics adopting this view is Clark's completion semantics [Cla78].

Example 37 Nonderivability versus finite failure.

![Diagram](image)

Figure 3.15: Positive loop.

From the HST of figure 3.15, without the rule $p \leftarrow p$, one may conclude $\vdash_{a} \neg p$. Adding a vacuous rule like $p \leftarrow p$ should not change the semantics and indeed it doesn’t if we interpret the negation as nonderivability. The derivation, with respect to target class $a$, of $\vdash \neg p$ depends on the failure of $\vdash p$. As the latter subgoal loops, it does not succeed and $\vdash \neg p$ is still obtained. A similar conclusion is produced via the well-founded model of the translated program.
\[ \vdash \neg p \leftarrow \not\vdash p \quad \vdash \neg p \leftarrow \not\vdash b p \]
\[ \vdash p \leftarrow \vdash \neg p \quad \vdash p \leftarrow \vdash \neg p \quad \not\vdash a \neg p \quad \vdash b p \leftarrow \vdash p. \]

\[ M = \{ \vdash \neg p, \not\vdash \neg p, \not\vdash p, \not\vdash p, \not\vdash a \neg p, \not\vdash b p \} \]

Using the interpretation of negation as finite failure the result is different. The goal \( \not\vdash p \) does not finitely fail thus hindering \( \vdash \neg p \) from success. A parallel situation occurs with \( \not\vdash \neg p \). So, a rule that adds nothing to the semantics ends up being able to prevent the inheritance of the default \( \neg p \).

The finite failure approach, though it is computationally more affordable, misses some conclusions obtained by the other method. This concerns specifically the positive loops, which cause infinite derivations and that turn out to be false in the well-founded semantics. The same problem does not show up with the negative loops, associated with cyclic derivations. In the well-founded semantics these redound to undefined atoms, opening the door to interpretations that minimize the false or the true outcomes. We choose the latter, for the strong atoms, and so we stay closer to the behavior of the finite failure method whose non-answer may be taken for a denial.

### 3.5.1 Explicitly modeling finite failure

Derivation procedures based on finite failure in some sense require an explicit or positive account for the derivation of negative goals. Under this spirit we present in figure 3.16 a set of inference rules where, instead of resorting to the nonderivability of certain goals, we explicitly define the weak finite failure relation. The symbol \( \vdash_a p \) means that \( p \) weakly (finitely) fails with respect to target \( a \).

The rules for the strong relation are essentially the same as in figure 3.1. The rules for the weak relation are reversed. The rule [WV], saying that an empty goal is always derived, disappears because such a goal must always fail with respect to the weak failure relation. In other words, there is no way to make \( \vdash \emptyset \) true. The rule for conjunction [WC] is now substituted by [FC], which may succeed in several ways because it makes a nondeterministic choice, i.e., if one property in a set belongs to the weak failure relation it renders the whole set weakly failed.

Failure through a reduction step is far more complex. The rule [WR] states how a weak derivation succeeds starting with a rule in any one of the target's superclasses. So the rule [FP] states that a property belongs to the weak failure relation if such derivation is confirmed in all those superclasses. As [WR] simultaneously
3.5 Reducing to finite failure

\[ [SV] \quad \frac{\| \neg_a \emptyset }{\| \neg_a \emptyset } \]

\[ [SC] \quad \frac{\| \neg_a p \quad \| \neg_a Q }{\| \neg_a p, Q } \]

\[ [SR'] \quad \frac{\| \neg_a Q \quad \| \neg_a \neg p }{\| \neg_a \emptyset \quad \{ p \leftarrow Q \in D_b \} } \]

\[ [FC] \quad \frac{\| \neg_a p }{\| \neg_a p, Q } \]

\[ [FP] \quad \frac{\| \neg_a^c p }{\| \neg_a p \quad \{ C = \{ a \mid a \leq b \} \} } \]

\[ [FSC] \quad \frac{\| \neg_a^* B \quad \| \neg_a^c p }{\| \neg_a^* p \quad \{ B = \{ Q \mid p \leftarrow Q \in D_b \} \} } \]

\[ [FA] \quad \frac{\| \neg_a^0 p }{\| \neg_a p } \]

\[ [FB] \quad \frac{\| \neg_a Q \quad \| \neg_a^* B }{\| \neg_a Q, B } \]

\[ [FE] \quad \frac{\| \neg_a^0 \emptyset }{\| \neg_a \emptyset } \]

\[ [FOV] \quad \frac{\| \neg_a b \quad \| \neg_a^{C \setminus C'} p }{\| \neg_a^c p \quad \{ C' = \{ x \mid b < x \} \} \quad \{ C \cap C' \neq \emptyset \} } \]

\[ [FLO] \quad \frac{\| \neg_a Q \quad \{ p \leftarrow Q \in D_b \} }{\| \neg_a p \quad \{ a \leq b \} } \]

Figure 3.16: Inference rules with finite failure.
requires the weak derivation of a rule body and the nonderivability of a local property there are two ways for the weak failure relation to succeed. One is if overriding takes place ([FOV]) and the other if all the available rules for the property in the class have a weakly failing body ([FSC]). The rule [FSC] takes care of the iteration on the classes and [FB] on the available bodies, for each class. Rules [FA] and [FE] are the base cases for [FSC]+[FOV] and [FB], respectively. Notice the universal quantification over all the rules in all the relevant classes, checking for their failure, that substitutes the negation of the existential quest of a successful rule that may serve as a derivation root.

An overriding done by a rule in a class b forces a weak failure in all the superclasses x of b. That’s why a single step of [FOV] may rule out several classes. We require that C ∩ C’ be non-empty to guarantee that each step represents a progress in the derivation. Rules [FOV] and [FLO], despite being declaratively correct, are non-constructive in the sense that in the normal sequence of application first comes [FOV], but the argument C’ depends on a b that is determined only after, in [FLO]. Rule [FLO] is basically equivalent to the former [LO] but it is presented in a slightly different way. The whole system is highly nondeterministic but an actual implementation has plenty of opportunities for optimization. For example, in [FLO] the search for rules should start from the lower classes, in order to maximize the set C’ of classes which are skipped by overriding.

An immediate advantage of the inference procedure with finite failure is that one derivation is represented by a single derivation tree, avoiding the extensive use of lateral subderivations.

\[
\frac{\text{[SR]}}{\frac{\text{[SV]}}{\frac{\text{[FSC]}}{\frac{\text{[FB]}}{\frac{\text{[FA]}}{\frac{\text{[FE]}}{\text{\neg_a \neg p}}}{\text{\neg_a \emptyset}}}{\text{\neg_a p}}}{\text{\neg_a^* p}}}{\text{\neg_a^* \{p\}}}{\text{\neg_a^* \emptyset}}}}{\text{\neg_a p}}}
\]

Figure 3.17: Derivation tree with finite failure.

The inference rules with finite failure more closely model the behavior of a Prolog-like system. See in figure 3.17 the reformulation of the derivation in example 37. Now there is a unified derivation tree, without the jumps between subtrees caused by checks of non-derivability. If it contains an infinite branch then the root goal is the blamed one. Using non-derivability, the infinite branch is encapsulated
in a subderivation and that lets the main derivation succeed.

**Lemma 90 (Weak failure soundness)** Every property in the weak failure relation belongs to the complement of the weak characterization, \( \vdash_a \subseteq \mathcal{P} \setminus \langle \sim a \rangle \).

**Proof:**

In this case, it is hard to establish a correspondence step-by-step. One may instead consider all the (incomplete) subderivations and jump from the set of their possible leaves to the root. From [FP] we know that \( \vdash_a p \) succeeds if there are no more classes where it could be weakly derived ([FA]). This means that each rule for \( p \) ([FSC]) either has a failing body or there is below it an overriding rule ([FOV]). With this hypothesis \( T^*(D\langle a \rangle) \) cannot contain \( p \) and so, by lemma 65, \( p \notin \langle \sim a \rangle \).

A step [FB] in the subderivation says that the corresponding rule has a failing body if one of the conjuncts \( p' \) is in \( \vdash_a \) ([FC]). Notice that the empty body \( \emptyset \) does not meet this criterion. By hypothesis this implies \( p' \notin \langle \sim a \rangle \). If there are no rules for \( p \) in a class, that branch of the subderivation ends with [FE], to which corresponds the trivial case of \( T_D \). The last case is when the leaf of the derivation is an application of [FLO]. If there is a rule \( p \leftarrow Q \) in a certain class \( b \) such that \( \vdash_a Q \) then \( p \) can be considered as weakly failing in all its superclasses. A step like this can be included in the derivation only if at least one class is affected, so that there is a progress in the derivation. If \( \vdash_a Q \) then, by hypothesis, \( Q \subseteq \langle a \rangle \) and so there is a stage \( k \) where it becomes a fact and \( T_D \) cannot use rules in any of its superclasses to obtain \( p \). So \( p \notin \langle \sim a \rangle \), i.e., \( p \in \mathcal{P} \setminus \langle \sim a \rangle \).

**Theorem 91 (Soundness FF)** Every strongly derived property, resorting to finite failure, belongs to the characterization, \( \vdash_a \subseteq \langle a \rangle \).

**Proof:**

The structure of the proof is an induction on the depth of the derivation complemented with a case analysis of the inference rules. The depth of a derivation is the maximum number of rules applied in its branches.

If the depth of the derivation is one then it must consist of an application of [SV]. The corresponding empty goal is \( \vdash_a \emptyset \). It also verifies \( \emptyset \subseteq \langle a \rangle \). This is the base case.

The induction step is of the form: if a condition being true for all the derivations of depth \( \leq n \) implies it is still true for depth \( n + 1 \), then it is true for all \( n \).

Rule [SC]: If \( \vdash_a p \) and \( \vdash_a Q \) then \( \vdash_a (p, Q) \). By hypothesis \( p \in \langle a \rangle \) and \( Q \subseteq \langle a \rangle \). Then obviously \( p, Q \subseteq \langle a \rangle \).
Rule \([\text{SR}]:\) If there is a rule \(p \leftarrow Q \in D_b\) such that \(a \leq b\), \(\models_a Q\) and \(\models_a \overline{p}\), then \(\models_a p\). To prove soundness we only need the result of lemma 90, \(\models_a \subseteq P \setminus <a>\) because any undue miss of weakly failing a property can only introduce incompleteness, not unsoundness. So, by the hypothesis, \(\overline{p} \in P \setminus <a>\) or, equivalently, \(p \not\in <a> = \leq <a> \cup T^*(D/ <a>)\). Noting, by (2.17), that \(J^n \subseteq <a>, \forall n\) and using lemma 54 this means that \(\overline{p}\) is not contained in \(T^*(D/J^{k'})\), for a sufficiently high \(k'\). Beyond \(k'\), either there is no support for \(\overline{p}\) or a rule for \(p\) becomes a fact in \(D/J^{k'}\), overriding the last rule for the opposite \(\overline{p}\) left uncovered.

Again by the hypothesis \(Q \subseteq <a>\). So, by lemma 61, the body of the above mentioned rule \(p \leftarrow Q \in D_b, a \leq b\) must become true at some stage \(k''\) in the iteration of (2.17), and some step in the inner iteration of (2.11). Then, for a stage \(k = \max\{k', k''\}\) the rule is active and not contradicted and so \(p \in \Delta_{D/Jk}\) because the restriction does not affect it, as \(\overline{p} \not\subseteq T^*(D/J^k)\), and therefore \(p \in <a>\).

There are problems with the completeness due to the intrinsic nonmonotonicity of the system. Having \(p \leftarrow p\) or equivalent in a class below a fact \(\overline{p}\) prevents the latter to be accepted in a top-down procedure based on finite failure, but has no effect in a bottom-up approach, if \(p\) cannot be concluded by other means.

### 3.5.2 Relative extensions

To clarify the relationship between the several derivation relations that were introduced we conclude the chapter with a few more results.

**Proposition 92 (Derivation relations)** The derivation relations with respect to a class \(a\) in a HST respect the following nested containment (see figure 3.18).

\[
\models_a \subseteq \models \subseteq \models_a \subseteq P \setminus \models_a
\]

**Proof:**

The first inclusion is obtained from theorems 91 and 86. The second comes from theorem 81. The third is deducible from lemma 90, which states \(\models_a \subseteq P \setminus <a>\), and the fact \(<a> \subseteq P\). Then \(<a> \subseteq P \setminus \models_a\). Noting that \(<a> = \models_a\), from theorems 88 and 87, we have the intended result.

So there may be a gap between \(\models_a\) and \(\models_a\), i.e., some properties that cannot be weakly derived do not belong to the weak finite failure relation either. These are the properties \(L\) depending on positive loops.

\[
P = \models_a \cup \models_a \cup L
\]

(3.33)
3.5 Reducing to finite failure

Figure 3.18: Relative size of the derivation relations.

Proposition 93 (Bounds on $\vdash_a$) The strong derivation relation depends simultaneously on the weak relation and on the negation of the complement of the weak relation,

$$\vdash_a \subseteq \vdash_a \cap \overline{\mathcal{P} \setminus \vdash_a} = \emptyset \vdash_a.$$

Proof:

The strong relation implies the weak by theorem 81. To obtain $\vdash_a p$, the inference rule [SV] requires $\vdash_a \overline{p}$, i.e., $\overline{p} \in \mathcal{P} \setminus \vdash_a$ or $p \in \mathcal{P} \setminus \vdash_a$. The equivalence between the intersection and the filtering of the weak derivation can be checked in figure 3.19, where $\overline{x}$ means the region labeled by $x$.

Figure 3.19: Bounding the strong derivation relation.
A similar relationship can be obtained for the finite failure relations.

\[ \models_{\alpha} \subseteq (\mathcal{P} \setminus \models_{\alpha}) \cap \overline{\models_{\alpha}} \] (3.34)

A last observation concerns pairs of complementary properties.

**Proposition 94 (Pairs)** If none or both of \( p \) and \( \overline{p} \) are weakly derived then none is strongly derived.

**Proof:**

If \( p, \overline{p} \notin \models_{\alpha} \) then, by the contrapositive of theorem 81 \( p, \overline{p} \notin \models_{\alpha}. \)

If \( p, \overline{p} \in \models_{\alpha} \) then \( p, \overline{p} \notin \mathcal{P} \setminus \models_{\alpha}. \) Taking the complement \( p, \overline{p} \notin \mathcal{P} \setminus \models_{\alpha}. \) So, by proposition 93, \( p, \overline{p} \notin \models_{\alpha}. \)
Chapter 4

Alternative views

In chapter 2 we presented a fixed-point bottom-up method of building the characterization. The inference system described in chapter 3 allows a goal-directed or top-down derivation of the conclusions it contains. We think it is clarifying to study the characterizations under a model-checking perspective.

4.1 Stable models of a class

4.1.1 Interpretations and models

A characterization of a given class in a HST is a set of properties. We want to define a notion of satisfaction that is aware of the overriding and neutralization phenomena and, at the same time, may be directly verified just looking at the relevant theories and the interpretation. For that, an interpretation must include, beyond the set of properties that corresponds to the characterization, an auxiliary set.

Definition 95 (Interpretation of a class) An interpretation of a class $a$ in a HST is a pair $(S, W)$ where $S$ and $W$ are sets of properties called respectively the strong and the weak components.

The intuitive meaning underlying the interpretation is that the set $S$ corresponds to the characterization and the set $W$ to the weak characterization.

Definition 96 (Model of a class) Let $D = (D_b)_{a \leq b}$ be the family of relevant class definitions for target class $a$. An interpretation $I = (S, W)$ of a class $a$ in a HST satisfies a rule $p \leftarrow Q \in D_b$ if and only if

1. $Q \subseteq S$, $\bar{p} \notin W \Rightarrow p \in S$, 

139
2. \( Q \subseteq W, (A \vdash p \leftrightarrow R \in D_c, c < b, R \subseteq S) \Rightarrow p \in W. \)

An interpretation \( I \) is a model of \( a \) if and only if it satisfies every rule in \( D \).

As can be concluded from the analysis of the negative conditions in the previous definition, the weak component is in charge of neutralization and the strong component is responsible for overriding. The positive conditions force a model to contain those consequences that are not subject to exceptions.

A definition of model usually establishes a sort of minimum requirements an interpretation should satisfy to be a model. In normal logic programs any interpretation extending beyond that minimum is also a model. In particular, the Herbrand base is a model of a logic program.

In HST, the correspondent to the Herbrand base is \( I = (P, P) \), which is always a model. As the notion of satisfaction is based on two sets negatively related, letting one of the sets grow beyond the “minimum” may allow the other to become too small. For instance, \( I = (\emptyset, P) \) is always a model, too. The situation is illustrated in example 38.

To enable the comparison of different models of a class, the order of definition 78 is adapted to the notion of interpretation.

**Definition 97 (Ordering models)** An interpretation \( I = (S, W) \) of a class \( a \) is less than or equal to an interpretation \( I' = (S', W') \) of the same class, written \( I \subseteq I' \), if and only if:

- \( S \subseteq S' \), or
- \( S = S' \) and \( W' \subseteq W \).

**Example 38** Several models of a simple HST.

\[
\begin{array}{|c|c|c|c|c|}
\hline
& I_1 & I_2 & I_3 & I_4 & I_5 \\
\hline S & p & & p, \neg p & \neg p & p, \neg p \\
W & p & p, \neg p & p & p, \neg p & p, \neg p \\
\hline
\end{array}
\]

Figure 4.1: The fact.

All the interpretations \( I_1 \) to \( I_5 \) are models. As there is no class below \( a \), \( p \) must belong to \( W \) in every model. If \( W \) does not contain \( \neg p \) then \( p \) must be in \( S \). This is the case of \( I_1 \). Interpretations \( I_3 \) and \( I_5 \) include some properties for which there
is no apparent justification in the HST but, as their components are superset of those in $I_1$, they are also models.

In $I_2$ the weak component is a superset of $W$ in $I_1$. However, as it contains $\neg p$, then $p$ is no longer required to be in the strong component, which is actually smaller than $S$ in $I_1$. The spurious property in $W$ is not as innocuous as it looks, because it frees $p$ to belong or not to the strong component. We have $I_2 \subseteq I_1$ and $I_4$ is not even comparable with $I_1$. □

From the example, we conclude that some models end up being not interesting. It is not only a matter of including some extra properties but that they may lack important ones. So we need further criteria to dismiss the less convenient models.

Due to the nonmonotonicity of HST adding unnecessary properties may be harmful. We introduce a notion of supported models, which are meant to contain just the properties for which there is a justification.

Definition 98 (Supported model) A model $M = (S, W)$ of a class is supported if and only if

1. $p \in S \Rightarrow \exists q \leftarrow Q \in D_b, Q \subseteq S, \overline{p} \not\in W$,

2. $p \in W \Rightarrow \exists q \leftarrow Q \in D_b, Q \subseteq W, (\exists \overline{p} \leftarrow R \in D_c, c < b, R \subseteq S)$.

Two other problems that are, to a certain extent, subsidiary of the unsupportedness are the inconsistency in the strong components and the clashes, i.e., properties that belong to the strong but not to the weak components. These problems are present in example 38, models $I_3$ and $I_5$. In that example, only $I_1$ is supported. The other have unjustified properties in at least one of the sets.

Although the class of supported models behaves a lot better than the set of all models, some problems remain.

Example 39 Unfounded property.

The symptoms of missing properties ($\neg p \not\in S_2, S_4$), inconsistency and clashes (see $I_3$) plague this example, despite all of $I_1$ to $I_4$ being supported models. The problem here, and it is a general difficulty of the model checking approach, is the positive loop $p \leftarrow p$. It masks $p$ as a supported property while actually it is just self-supported, or unfounded, and by this reason set to false by the bottom-up semantics. □

Positive loops are easily detectable by static analysis of the dependency graph induced by the rules. Whether the loop constitutes an unfounded set is not so easy to see because there may be independent support for some of the properties.
In the case of definite programs in Logic Programming, all the models beyond the least model contain the atoms in it and possibly some unsupported atoms. Supported models of a HST, in general, can be \( \sqsubseteq \)-smaller or greater than the other models. In particular, the \( \sqsubseteq \)-least model \( \langle \emptyset, \mathcal{P} \rangle \) is not necessarily supported.

### 4.1.2 Stability operator

In this section we study a restricted class of models which possess a stability quality.

The consequences of a definite program can be clearly stated as the least fixed-point of an immediate consequence operator. A general program allows negative literals in rule bodies, thus introducing nonmonotonicity. To deal with it and still profit from the simplicity of positive programs, the stable model semantics [GL88] proposed to settle the interpretation of negative literals before computing the consequences of the positive part of the program. So, an interpretation must be known beforehand, which is used to remove all the negative literals from the clauses. If the least fixed-point of the immediate consequence operator associated to the simplified program coincides with the given interpretation it is a stable model.

In HST, exceptions and ambiguities represent two forms of nonmonotonicity. Building on the idea of settling the negative part before computing the positive consequences, we define a stability operator in two steps. The first captures the overriding and the second the neutralization. Instead of defining transformations on the program, we prefer to redefine the immediate consequence operator by adding the appropriate negative preconditions.

**Definition 99 (Stability operator)** The weak (strong) immediate consequence operator

\[
T_D^w, T_D^s : \varphi(\mathcal{P}) \times \varphi(\mathcal{P}) \rightarrow \varphi(\mathcal{P})
\]

takes a strong (weak) component and a set of properties and gives a weak (strong)
4.1 Stable models of a class

(component,)

\[ T^w_D(S)(J) = \{ p \mid p \leftarrow Q \in D_b, Q \subseteq J, \ A \overline{p} \leftarrow R \in D_c, c < b, R \subseteq S \} \] (4.1)

\[ T^w_D(W)(J) = \{ p \mid p \leftarrow Q \in D_b, Q \subseteq J, \overline{p} \notin W \}. \] (4.2)

The stability operator

\[ \Sigma_D : \varphi(P) \times \varphi(P) \rightarrow \varphi(P) \times \varphi(P) \]

from interpretations to interpretations is defined by

\[ \Sigma_D((S,W)) = (T^w_D(W) \uparrow \omega, T^w_D(S) \uparrow \omega). \] (4.3)

The strong component \( S \) fixes, in a single step, the respective consequences in all the theories. After having thus settled the overriding expressed in the negative part of \( T^w_D \), the least fixed-point of its positive part is computed. \( T^w_D \) is equivalent to the common immediate consequence operator applied to the union of the theories without the rules overridden by \( S \). It gives all the weak properties, which have some reason to exist, whether ambiguous or not.

A weak component \( W \) fixes the negative part of \( T^w_D \). The strong consequences are the least fixed-point of the positive part of \( T^w_D \). Again, \( T^w_D \) is like the common immediate consequence operator but applied to the union of the theories without the rules neutralized by \( W \).

Finally, if the resulting components are equal to the initial ones, then we have that \( I = (S, W) \) is a stable model.

Definition 100 (Stable model) An interpretation \( I \) is a stable model if and only if

\[ I = \Sigma_D(I). \]

Lemma 101 (Weak monotony) The weak immediate consequence operator is monotonic on its second argument,

\[ J_1 \subseteq J_2 \Rightarrow T^w_D(S)(J_1) \subseteq T^w_D(S)(J_2). \]

Proof:

Let us suppose that \( p \in T^w_D(S)(J_1) \) and

\[ p \notin T^w_D(S)(J_2). \] (4.4)

By 4.1, there is a rule \( p \leftarrow Q \in D_b \) such that \( Q \subseteq J_1 \) and

\[ A \overline{p} \leftarrow R \in D_c, c < b, R \subseteq S. \] (4.5)

But, from (4.4), every rule \( p \leftarrow Q \in D_b \) that verifies (4.5) must have \( Q \subseteq J_2 \), contradicting the hypothesis \( J_1 \subseteq J_2 \).
Lemma 102 (Strong monotony) The strong immediate consequence operator is monotonic on its second argument,

\[ J_1 \subseteq J_2 \Rightarrow T^S_B(W)(J_1) \subseteq T^S_B(W)(J_2). \]

Proof:

Let us suppose that \( p \in T^S_B(W)(J_1) \) and

\[ p \notin T^S_B(W)(J_2). \tag{4.6} \]

By 4.2, there is a rule \( p \leftarrow Q \in D_b \) such that \( Q \subseteq J_1 \) and

\[ \overline{p} \notin W. \tag{4.7} \]

But, from (4.6), every rule \( p \leftarrow Q \in D_b \) that verifies (4.7) must have \( Q \subseteq J_2 \), contradicting the hypothesis \( J_1 \subseteq J_2 \).

Lemma 103 (Weak anti-monotony) The weak immediate consequence operator is anti-monotonic on its first argument,

\[ S_1 \subseteq S_2 \Rightarrow T^w_B(S_1)(J) \supseteq T^w_B(S_2)(J). \]

Proof:

Let us suppose that \( p \in T^w_B(S_2)(J) \) and

\[ p \notin T^w_B(S_1)(J). \tag{4.8} \]

By 4.1, there is a rule \( p \leftarrow Q \in D_b, Q \subseteq J \) such that, for every rule \( \overline{p} \leftarrow R \in D_c, c < b \) we have \( R \subseteq S_2 \). But then, from (4.8), there is a rule \( \overline{p} \leftarrow R \in D_c, c < b \) with \( R \subseteq S_1 \), contradicting the hypothesis \( S_1 \subseteq S_2 \).

Lemma 104 (Strong anti-monotony) The strong immediate consequence operator is anti-monotonic on its first argument,

\[ W_1 \subseteq W_2 \Rightarrow T^S_B(W_1)(J) \supseteq T^S_B(W_2)(J). \]

Proof:

Similar to the proof of lemma 103.

Corollary 105 (lfp) The operators \( \lambda S.T^w_B(S) \uparrow \omega \) and \( \lambda W.T^S_B(W) \uparrow \omega \) are anti-monotonic.
4.1 Stable models of a class

Proof:

We want to prove that

$$S_a \subseteq S_b \Rightarrow T_D^\omega(S_a) \uparrow \omega \supseteq T_D^\omega(S_b) \uparrow \omega.$$  

By lemma 103, with $S_a \subseteq S_b$, we have $J^1_a = T_D^\omega(S_a)(\emptyset) \supseteq T_D^\omega(S_b)(\emptyset) = J^1_b$. Assuming $J^i_a \supseteq J^i_b$,

$$J^{i+1}_a = T_D^\omega(S_a)(J^i_a) \supseteq T_D^\omega(S_b)(J^i_b) = J^{i+1}_b$$ by lemma 103

As $T_D^\omega(S)$ is monotonic and finitary it has a least fixed-point obtainable in $\omega$ steps, $T_D^\omega(S_a) \uparrow \omega = J^\omega_a \supseteq J^\omega_b = T_D^\omega(S_b) \uparrow \omega$.

The proof for the case $\lambda W.T_D^\omega(W) \uparrow \omega$ is similar.

The left (right) component of $\Sigma_D(I)$ is a function of the right (left) component of $I$. The sequence of interpretations generated by an iterative process of $\Sigma_D$ has the characteristic that the two lines of dependences, which begin respectively with the left and right components of the first interpretation, do not interfere. So, the left components of even order are related but independent of the left components of odd order.

Proposition 106 (Squared operator) The squared stability operator is monotonic on both its arguments.

Proof:

Let $I_i = (S_i, W_i)$ and $I''_i = (S''_i, W''_i) = \Sigma_D^2(I_i)$. Then we want to prove that

$$S_1 \subseteq S_2 \Rightarrow S''_1 \subseteq S''_2$$ and $$W_1 \subseteq W_2 \Rightarrow W''_1 \subseteq W''_2.$$

If $S_1 \subseteq S_2$ then, by lemma 103,

$$W'_i = T_D^\omega(S_1) \uparrow \omega \supseteq T_D^\omega(S_2) \uparrow \omega = W'_2$$

and, by lemma 104,

$$S''_1 = T_D^\omega(W'_1) \uparrow \omega \subseteq T_D^\omega(W'_2) \uparrow \omega = S''_2.$$

The proof for the weak components is similar.

The previous proposition does not guarantee the monotonicity of $\Sigma_D^2$ under the order $\subseteq$. It may happen that $I_1 \subseteq I_2$ and $I''_1 = \Sigma_D^2(I_1) \not\subseteq \Sigma_D^2(I_2) = I''_2$ if $S_1 \subset S_2, W_1 \subset W_2$ and $S''_1 = S''_2, W''_1 \subset W''_2$. 
The stability operator is more interesting in the case the strong and weak components of an interpretation are related. If we impose on the initial interpretation the condition
\[ \mathcal{W}_1 = T^w_B(S_1) \uparrow \omega \] (4.9)
then the following odd iterations respect the same condition and the even iterations respect its dual
\[ S_2 = T^s_B(W_2) \uparrow \omega. \] (4.10)

Notice that, for \( n > 0 \),
\[ S_{2n+1} = S_{2n} \quad \text{and} \quad W_{2n} = W_{2n-1}, \] (4.11)
i.e., each iteration of \( \Sigma_D \) performs alternately a step in just one of the components.

**Theorem 107 (Monotony of \( \Sigma_D \))** Let \( \mathcal{I}_1 = (S_1, W_1) \) be an interpretation such that \( W_1 = T^w_B(S_1) \uparrow \omega \). Then \( \mathcal{I}_k \) defined by \( \mathcal{I}_{k+1} = \Sigma^k_D(\mathcal{I}_1) \), \( k \geq 0 \) is a monotonic sequence.

**Proof:**

Let us suppose that \( \mathcal{I}_1 \subseteq \mathcal{I}_2 \). Then \( S_1 \subseteq S_2 \). As, by definition 99, \( W_1 = T^w_B(S_1) \uparrow \omega = W_2 \) we have \( S_2 = T^s_B(W_1) \uparrow \omega = T^s_B(W_2) \uparrow \omega = S_3 \). Then \( S_1 \subseteq S_3 \). By proposition 106, \( (S_k) \) is a monotonic non-decreasing sequence.

As \( W_1 = T^w_B(S_1) \uparrow \omega \) and \( W_3 = T^w_B(S_2) \uparrow \omega \) and we are assuming \( S_1 \subseteq S_2 \) then, by lemma 103, \( W_1 \supseteq W_3 \). Using again proposition 106, \( (W_k) \) is a monotonic non-increasing sequence. Therefore, \( \mathcal{I}_k \subseteq \mathcal{I}_{k+1} \) for all \( k > 0 \) and \( (\mathcal{I}_k) \) is a monotonic non-decreasing sequence.

The proof for the case \( \mathcal{I}_2 \subseteq \mathcal{I}_1 \) follows the same steps. 

A similar result holds if the condition on the initial interpretation is the dual \( S_1 = T^s_B(W_1) \uparrow \omega \). So, \( \Sigma_D \) is monotonic if its domain is restricted to the interpretations respecting (4.9) and (4.10). It is also finitary. Therefore it has a least fixed-point obtainable in \( \omega \) steps, \( \Sigma_D \uparrow \omega \), where \( \Sigma^0_D = \mathcal{I}_0 = (\emptyset, P) \). Notice that \( \mathcal{I}_1 = \Sigma_D(\mathcal{I}_0) = (\emptyset, T^w_B(\emptyset) \uparrow \omega) \) agrees with the condition of theorem 107.

**Proposition 108 (Supportedness)** The fixed-points of the stability operator are supported models.

**Proof:**

From definition 100, \( (S, W) = \Sigma_D((S, W)) \). Using (4.3), (4.1) and (4.2),
\[ W = T^w_B(S) \uparrow \omega = T^w_B(S)(W) \]
\[ S = T^s_B(W) \uparrow \omega = T^s_B(W)(S). \]
The result follows from the coincidence between (4.1) and (4.2) and definitions 96 and 98.

The converse statement is not true, i.e., there are supported models that are not stable models. This is the case of models containing unfounded sets in their strong or weak components, which are stripped off by the bottom-up nature of the positive parts of the stability operator. Check for instance, in figure 4.2, that only $\mathcal{I}_1$ is a stable model. All the other contain $p$, whose sole rule depends on itself.

To investigate the structure of the class of stable models we define a binary operation on interpretations. It is meant to give the common properties so it has a flavor of intersection. However, the weak components, due to its negative role, are unified instead.

**Definition 109 (Intersection)** The intersection of interpretations $\mathcal{I}_1 = (\mathcal{S}^1, \mathcal{W}^1)$ and $\mathcal{I}_2 = (\mathcal{S}^2, \mathcal{W}^2)$ is the interpretation $\mathcal{I} = (\mathcal{S}, \mathcal{W}) = \mathcal{I}_1 \cap \mathcal{I}_2$ such that

\[
\mathcal{S} = \mathcal{S}^1 \cap \mathcal{S}^2 \\
\mathcal{W} = \mathcal{W}^1 \cup \mathcal{W}^2.
\]

The intersection of two stable models is not always stable and sometimes even not a model.

**Example 40** An intersection that is not a model.

In figure 4.3, the models $M_1$-$M_4$ are all stable. The intersection

\[
\mathcal{I} = M_2 \cap M_3 = \left\{ \neg t \right\} \cup \left\{ r, \neg r, p, \neg p, \neg t \right\}
\]

is not a model because $r, p \in \mathcal{W}$ while $t \notin \mathcal{W}$, as it does not belong to $M_2$ or $M_3$ individually considered. The problem is not completely overcome by adding $t$ to $\mathcal{W}$ because then $\neg t$ becomes unsupported in $\mathcal{S}$. If $\neg t$ is removed, the resulting interpretation is a stable model. Thus, the interplay between $\mathcal{S}$ and $\mathcal{W}$ stops here, in this example.

The changes in $\mathcal{S}$ and $\mathcal{W}$ are monotonic and reinforce each other, i.e., an extra weak property required, for instance, by a rule simultaneously containing properties in $\mathcal{W}^1$ and $\mathcal{W}^2$, may neutralize a strong property, which becomes unsupported. Purging the strong property, on its turn, may cancel an overriding and make a rule weakly not satisfied, requiring a new addition to $\mathcal{W}$. As the number of properties is finite the process must finish. All along it, $\mathcal{S}$ goes on satisfying $D$, because only unsupported properties are purged, and $\mathcal{W}$ remains supported, because only heads of rules that are not satisfied are added.
As $S$ can only become smaller and $W$ bigger, the result of the adjustment is $M_1 \subseteq I$.

However, the class of stable models enjoys the property that there is always a stable model that is $\subseteq$-less than the intersection of two stable models.

**Proposition 110 (Intersecting)** Let $I$ be the intersection of two stable models. There is a stable model $M$ such that $M \subseteq I$.

**Proof:**

Let $M_1 = \langle S^1, W^1 \rangle = \Sigma_D(M_1)$ and $M_2 = \langle S^2, W^2 \rangle = \Sigma_D(M_2)$ be stable models. Let $I = M_1 \cap M_2 = \langle S_1 \cap S_2, W_1 \cup W_2 \rangle$ and $\Sigma_D(I) = \langle S, W \rangle$.

Then, by lemmas 104 and 103,

\[
S = T_B(W_1 \cup W_2) \uparrow \omega \subseteq T_B(W_1) \uparrow \omega = S_1
\]

\[
S \subseteq T_B(W_2) \uparrow \omega = S_2
\]

\[
W = T_B(S_1 \cap S_2) \uparrow \omega \supseteq T_B(S_1) \uparrow \omega = W_1
\]

\[
W \supseteq T_B(S_2) \uparrow \omega = W_2,
\]

i.e., $S \subseteq S_1 \cap S_2$, $W \supseteq W_1 \cup W_2$. Therefore $\Sigma_D(I) \subseteq I$.

Iterating $\Sigma^k_D(I)$ we get, by proposition 106, a monotonic non-increasing sequence on the strong component and a monotonic non-decreasing sequence on the
4.1 Stable models of a class

weak component. As the operator is finitary, a fixed-point is eventually reached which is \( \subseteq \)-less than \( I \).

The stability operator can be used as an alternative to the operator \( S_D \). Starting with the interpretation \( \langle \emptyset, P \rangle \) and iterating \( \Sigma_D \) we obtain a sequence of models, possibly not supported, with a least fixed-point equal to \( \langle <a>, \overline{a} \rangle \).

**Lemma 111 (Strong is weak)** Let \( (S^{2^k}, W^{2^k}) = \Sigma^{2^k}_D(\emptyset, P) \) and \( J^k = S_D^k(\emptyset) \). Assuming \( S^{2^k} = J^k \) and \( W^{2^k} = J^{k-1} \cup T^*(D/J^{k-1}) \) then \( S^{2^k} \subseteq W^{2(k+1)} \) holds.

**Proof:**

We will prove that for every \( n \)

\[
I_n = T_D^n(W^{2^k}) \uparrow n \subseteq J_n = T_D^n(S^{2^k}) \uparrow n.
\]

Check with (4.1), (4.2) and (4.11) that for \( n = \omega \) that is the intended result.

**Base case.** \( I_0 = \emptyset = J_0 \).

**Inductive step.** \( p \in I_{n+1} \). From (4.2) there is a rule

\[
p \leftarrow Q \in D_b, Q \subseteq I_n, \overline{p} \notin W^{2^k}.
\]

We suppose that class \( b \) is a minimal class enjoying \( Q \subseteq S^{2^k} \).

Assuming, by contradiction, that \( p \notin J_{n+1} \) then, by (4.1) and as, by hypothesis, \( Q \subseteq J_n \), there is a rule

\[
\overline{p} \leftarrow R \in D_c, c < b, R \subseteq S^{2^k}.
\]

But from (2.20), \( S^{2^k} \subseteq W^{2^k} \) and so \( R \subseteq W^{2^k} \). From (4.1),

\[
W^{2^k} = T_D^{\overline{p}}(S^{2(k-1)})(W^{2^k})
\]

implying that either \( \overline{p} \in W^{2^k} \), contradicting (4.12), or there is a rule \( p \leftarrow Q' \) below \( c \) with \( Q' \subseteq S^{2(k-1)} \subseteq S^{2^k} \), because from (2.16) \( J^{k-1} \subseteq J^k \), and so class \( b \) is not minimal. Therefore \( p \in J_{n+1} \) and \( S^{2^k} \subseteq W^{2(k+1)} \).

**Theorem 112 (Stable is safe)** Let \( D \) be the family of relevant theories with respect to a target class \( a \). The least fixed-point of the stability operator coincides with the conclusions of the bottom-up iteration,

\[
\Sigma_D \uparrow \omega = \langle <a>, \overline{a} \rangle.
\]
Proof:

Actually we prove the stronger statement that the equivalence holds at every step \( k \) of the bottom-up iteration. Let \( \mathcal{I}^{k} = (\mathcal{S}^{k}, \mathcal{W}^{k}) \), with \( \mathcal{I}^{0} = (\emptyset, \mathcal{P}) \). Then, for \( k > 0 \) we prove that

\[
\mathcal{S}^{2k} = J^{k}, \quad \mathcal{W}^{2k} = J^{k-1} \cup T^{*}(D/J^{k-1}).
\]

**Base case.** We have \( \mathcal{S}^{0} = \emptyset = J^{0} \) and \( \mathcal{S}^{1} = T_{D}^{*}(\mathcal{P}) \uparrow \omega = \emptyset \). So, \( \mathcal{W}^{2} = \mathcal{W}^{1} \).

From (4.3) and (4.1), we get the following fixed-point equation

\[
\mathcal{W}^{2} = T_{D}^{*}(\mathcal{S}^{0}) \uparrow \omega = T_{D}^{*}(\mathcal{S}^{0})(\mathcal{W}^{2})
\]

\[
= \{ p \mid p \leftarrow Q \in D_{b}, Q \subseteq \mathcal{W}^{2}, \neg p \leftarrow R \in D_{c}, c < b, R \subseteq \mathcal{S}^{0} \}
\]

\[
= \{ p \mid p \leftarrow Q \in D_{b}, Q \subseteq \mathcal{W}^{2}, \neg p \leftarrow R \in D_{c}, c < b \}.
\]

Comparing this equation with (2.11), we conclude that \( \mathcal{W}^{2} \) is also the least fixed-point of \( T_{D} \),

\[
\mathcal{W}^{2} = T_{D}(\mathcal{W}^{2}) = T^{*}(D) = J^{0} \cup T^{*}(D/J^{0}).
\]

\[
\mathcal{S}^{2} = T_{D}^{*}(\mathcal{W}^{2}) \uparrow \omega
\]

\[
= \{ p \mid p \leftarrow Q \in D_{b}, Q \subseteq \mathcal{S}^{2}, \neg p \notin \mathcal{W}^{2} \}
\]

\[
= \{ p \mid p \leftarrow Q \in D_{b}, Q \subseteq \mathcal{S}^{2}, \neg p \notin T^{*}(D) \}
\]

\[
= \Delta_{D}
\]

by (2.14) and noting that the condition of absence of contradictory facts below \( b \) in (2.11) is subsumed by \( \neg p \notin T^{*}(D) \).

**Inductive step.** In step \( k \) and using (4.3) and (4.1), we have the following fixed-point equation

\[
\mathcal{W}^{2(k+1)} = T_{D}^{w}(\mathcal{S}^{k}) \uparrow \omega = T_{D}^{w}(\mathcal{S}^{k})(\mathcal{W}^{2(k+1)})
\]

\[
= \{ p \mid p \leftarrow Q \in D_{b}, Q \subseteq \mathcal{W}^{2(k+1)}, \neg p \leftarrow R \in D_{c}, c < b, R \subseteq \mathcal{S}^{2k} \}.
\]

By hypothesis, it becomes

\[
\mathcal{W}^{2(k+1)} = \{ p \mid p \leftarrow Q \in D_{b}, Q \subseteq \mathcal{W}^{2(k+1)}, \neg p \leftarrow R \in D_{c}, c < b, R \subseteq J^{k} \}.
\]

From lemma 111 \( J^{k} = \mathcal{S}^{2k} \subseteq \mathcal{W}^{2(k+1)} \) and we may divide, using (2.15), the theories \( D \) by \( J^{k} \) to get

\[
(\mathcal{W}^{2(k+1)} \setminus J^{k}) = \{ p \mid p \leftarrow Q' \in (D/J^{k})_{b}, Q' \subseteq (\mathcal{W}^{2(k+1)} \setminus J^{k}), \neg p \leftarrow \in (D/J^{k})_{c}, c < b \},
\]
where \( Q' \) is the subset of properties in \( Q \) that are not in \( J^k \). Comparing with (2.11), \( \mathcal{W}^{2(k+1)} \setminus J^k \) is a fixed-point of \( T_{D/J^k} \),

\[
\mathcal{W}^{2(k+1)} \setminus J^k = T_{D/J^k}(\mathcal{W}^{2(k+1)} \setminus J^k) = T^*(D/J^k).
\]

As \( J^k \subseteq \mathcal{W}^{2(k+1)} \) we have \( \mathcal{W}^{2(k+1)} = J^k \cup T^*(D/J^k) \).

\[
S^{2(k+1)} = T_D(\mathcal{W}^{2(k+1)}) \upharpoonright \omega = T_D(\mathcal{W}^{2(k+1)})(S^{2(k+1)})
\]

\[
= \{ p \mid p \leftarrow Q \in D_b, Q \subseteq S^{2(k+1)}, \bar{p} \notin \mathcal{W}^{2(k+1)} \}.
\]

Dividing the family of theories \( D \) by \( J^k \) and as \( S^k \) is a monotonically non-decreasing sequence, we get

\[
S^{2(k+1)} \setminus J^k =
\]

\[
= \{ p \mid p \leftarrow Q' \in (D/J^k)_b, Q' \subseteq (S^{2(k+1)} \setminus J^k), \bar{p} \notin \mathcal{W}^{2(k+1)} \setminus J^k \}
\]

\[
= \{ p \mid p \leftarrow Q' \in (D/J^k)_b, Q' \subseteq (S^{2(k+1)} \setminus J^k), \bar{p} \notin T^*(D/J^k) \}
\]

\[
= \Delta_{D/J^k},
\]

by (2.11) and (2.13).

So, by (2.16), \( S^{2(k+1)} = J^k \cup \Delta_{D/J^k} = J^{k+1} \).
One of its main conclusions is the similarity, at this level of abstraction, between several major approaches to commonsense and nonmonotonic reasoning. It will be shown below that also HST belongs to this family. All these systems have an overall structure of arguments which may attack and counterattack each other.

**Definition 114 (Acceptability)** An argument $A$ is acceptable with respect to a set of arguments $S$ if each of the arguments attacking $A$ is also attacked by elements of $S$.

A rational agent, put before an argumentation, will choose a set of arguments $S$ which are not attacking each other. Such a set is called conflict-free and does not contain any two arguments $A, B$ such that $A$ attacks $B$. Moreover, a rational agent will not leave out of $S$ any argument he can accept with respect to $S$.

**Definition 115 (Complete extension)** A conflict-free set of arguments $S$, whose elements are all acceptable with respect to $S$, is called a complete extension if and only if each non-included argument is not acceptable with respect to the set.

This definition suggests a way to obtain a complete extension, starting from a conflict-free set of arguments, by the successive computation of the acceptable arguments.

**Definition 116 (Characteristic function)** Given an argumentation framework $AF = (AR, attacks)$, its characteristic function, from the power set of arguments $AR$ into itself, is $F_{AF}(S) = \{A \mid A$ is acceptable wrt $S\}$.

The set of conflict-free fixed points of $F_{AF}$ coincides with the set of complete extensions and constitutes a complete semilattice, with respect to set inclusion. The operator $F_{AF}$ is monotonic. Its least fixed point, which is also the least complete extension, is called grounded extension and denoted by $GE_{AF}$. It is the least among the sets that a rational agent could adopt and so it defines the skeptical semantics of the argumentation framework.

The definition of acceptability is a version of the aphorism "The enemies of our enemies are our friends". In a chain of successive attacks ending on $A$, those arguments which are an odd number of steps apart from it play the role of potential attackers of $A$, while those separated by an even number of steps are indirect defenders. An argument which, at the same time, potentially attacks and indirectly defends $A$ is said to be controversial.

A controversial argument is puzzling by itself, but the situation gets worse when the relation $attacks$ contains cycles. If the number of arcs in the cycle is odd then
an argument $A$ in the cycle potentially attacks itself, because there is an odd chain from $A$ to $A$, and also indirectly defends itself, along an even chain traversing the cycle twice. So, every argument in the cycle is controversial with respect to itself. In the case of an even cycle, each argument is a non-controversial indirect defender of itself, a much more understandable situation.

A cycle is isolated if its nodes are not attacked by arguments outside the cycle. The attacks and counterattacks in an isolated even cycle have all the same strength and so the corresponding arguments do not belong to the grounded extension. The simplest case (see example 41) is $\text{AF}_1 = \langle \{A, B\}, \{(A, B), (B, A)\} \rangle$ which has an empty grounded extension. However, a rational agent would not betray his principles if, in front of a balanced discussion, he decides to choose one side. Assuming a sequential numbering of the nodes in the cycle, he may pick up all the even or, alternatively, all the odd numbered nodes, provided that he carries along also every argument which may become acceptable (see $\text{AF}_2$), in order to keep up with a complete extension. The sets $E_1 = \{A\}$ and $E_2 = \{B\}$ are complete extensions of $\text{AF}_1$ and can be viewed as plausible alternatives. The set of maximal complete extensions defines the credulous semantics of an argumentation framework.

The same rational agent would get trapped, should he try to apply the previous method to an isolated odd cycle, because it is not possible to add all the even, or all the odd, numbered nodes without running into conflicts. Leaving out two consecutive nodes would render the next one not acceptable and the successive application of $\text{FAF}$ would never reach a fixed point. The simplest example of this situation is the loop with three nodes (see $\text{AF}_3$).

The simplest controversial framework is $\text{AF}_4 = \langle \{A\}, \{(A, A)\} \rangle$ whose sole complete (and grounded) extension is empty.

**Definition 117 (Stable extension)** A conflict-free set of arguments $S$ is a stable extension iff $S$ attacks each argument which does not belong to $S$.

Some controversial argumentation frameworks do not possess stable extensions (see $\text{AF}_4$, $\text{AF}_5$). Clearly, the grounded extension, which always exists, may be not stable.

**Theorem 118 (Non-controversial AF [Dun93b])** If an argumentation framework is not controversial then:

a) each maximal complete extension is stable and.

b) there is at least one stable extension.

**Example 41** Argumentation frameworks with cycles.
<table>
<thead>
<tr>
<th>$AF$</th>
<th>$AF_1$</th>
<th>$AF_2$</th>
<th>$AF_3$</th>
<th>$AF_4$</th>
<th>$AF_5$</th>
<th>$AF_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$GE_{AF}$ (skeptical)</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${A, C, E}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${B, D}$</td>
</tr>
<tr>
<td>maximal complete extensions (credulous)</td>
<td>${A}$</td>
<td>${A, C, F}$</td>
<td>${A, C, E}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>${B, D}$</td>
</tr>
<tr>
<td>stable extensions</td>
<td>${A}$</td>
<td>${A, C, F}$</td>
<td>${A, C, E}$</td>
<td>$\text{no}$</td>
<td>$\text{no}$</td>
<td>${B, D}$</td>
</tr>
<tr>
<td>controversial</td>
<td>$\text{no}$</td>
<td>$\text{no}$</td>
<td>$\text{no}$</td>
<td>$\text{yes}$</td>
<td>$\text{yes}$</td>
<td>$\text{yes}$</td>
</tr>
</tbody>
</table>

Figure 4.4: Simple cycles.

In the directed graphs of figure 4.4, nodes represent arguments and there is an arc from node $A$ to node $B$ iff $A$ attacks $B$.

Comments:

$AF_1$ $\{A\}$ is conflict-free, $A$ is acceptable with respect to $\{A\}$, and $B$ is not. So, $\{A\}$ is a complete extension. The same for $\{B\}$.

$AF_2$ Though only $A$ and $C$ are odd nodes in the cycle, after choosing them, $F$ becomes acceptable and must be included in the same complete extension.

$AF_3$ Non-isolated cycle. $F_{AF_5}(\emptyset) = \{E\}$, $F_{AF_5}(\{E\}) = \{E, A\}$, $F_{AF_5}(\{E, A\}) = \{E, A, C\} = GE_{AF_5}$. The external attack $E \rightarrow D$ unbalances the situation in $AF_2$ in favor of $\{A, C, E\}$.

$AF_4$ Although $A$ is acceptable with respect to $\{A\}$, $\{A\}$ is not conflict-free and so it cannot be a complete extension.

$AF_5$ $F_{AF_5}(\{A\}) = \{C\}$, $F_{AF_5}(\{C\}) = \{B\}$, $F_{AF_5}(\{B\}) = \{A\}$. Unstable.

$AF_6$ $F_{AF_6}(\emptyset) = \{D\}$, $F_{AF_6}(\{D\}) = \{D, B\} = GE_{AF_6}$. Controversial $B$ belongs to the extension.

Notice that, like the skeptical semantics, the credulous semantics always exists. On the contrary, there may be no stable models in some controversial frameworks. Their main interest lies in facilitating comparisons with other systems.

This short presentation of the Argumentation Theory finishes here. More details can be found in [Dun93b].
4.2.2 Embedding HST in the Argumentation Framework

In order to show how HST can be embedded in the argumentation framework we must state what an argument is and when does an argument attack another. An argument is about a target class enjoying a property. It will encapsulate a line of reasoning behind that conclusion, disregarding possible interactions with other conclusions and any concerns with specificity in each step. It is enough to find a set of rules in the target's superclasses which, starting from facts, supports the conclusion. The following definition applies only to a restricted class of HSTs. We will give below a general definition.

**Definition 119 (Embedding, rest.)** In the context of a HST $\mathcal{H} = \langle \mathcal{C}, \mathcal{P}^+, \prec, \mathcal{D} \rangle$, let $a \in \mathcal{C}$ be a target class and $S \subseteq \mathcal{C} \times \mathcal{P}$. Then an argument for a property $q$ using a rule from a class $b$ is a pair $(S, b\rightarrow q)$ satisfying the following conditions:

1. $S$ is under an irreflexive partial order with a greatest element $b\rightarrow q$, called the head of the argument,

2. if $S_0$ is the set of immediate predecessors of $c\rightarrow p$ in $S$ then there is a rule $p \leftarrow \{p' \mid c' \rightarrow p' \in S_0\}$ in $\mathcal{D}_c$, with $a \leq c$, and

3. there is at most one pair $c\rightarrow p$ in $S$ for each property $p$.

The argumentation framework for $\mathcal{H}$, with respect to target $a$, is

$$AF(\mathcal{H}) = \langle AR_\mathcal{H}, \text{attacks}_\mathcal{H} \rangle$$

where $AR_\mathcal{H}$ is the set of arguments and $(S, c\rightarrow p) \text{attacks}_\mathcal{H} (S', c'\rightarrow p')$ if and only if there is a class $c' \neq c$ such that $c''\rightarrow p \in S'$.

At first sight one could think of arguments being just properties for classes. However, that simpler notion would not be expressive enough to allow the definition of an attack relation with the power of undermining the justification of a conclusion, but only the conclusion itself. Remember that attack is just a relation between arguments. This would leave the door open to accept an argument while refusing its subarguments. By this reason, the argument must code complete reasonings, i.e. a conclusion and all the subconclusions supporting it, as in HST every conclusion about properties is attackable. Defeating a conclusion implies attacking all the arguments which rely on it.

The arguments abstract out the forward chaining part of reasoning in HST, which generates potential conclusions (defaults) in every possible way. The nonmonotonic part, i.e. avoiding contradictions taking into account specificity, is left for the attack relation. The great appeal of the argumentation theory comes from this judicious separation which gives the right level of abstraction to analyze several proposals for nonmonotonic reasoning.
4.2.2.1 Arguments

Taking a closer look at definition 119 we see that it shapes the arguments very tightly. The set $S$ contains class-property pairs instead of simple properties because, when specificity comes into play, knowing the properties' source classes is crucial. Condition 2 simultaneously defines, recursively, when does a pair $c$-$p$ belong to $S$ and constrains the partial order in $S$ to be the transitive closure of the direct support relation from the properties in a rule's body to the respective head. The base of the recursion consists of the facts (rules with empty bodies). This condition insures a certain form of minimality when the HST contains different rules for the same property in the same class. It could seem that the union of two arguments for the same property, but differing in the choice for such rules, would be also an argument. That is not true, as can be seen in example 42.

The union of two arguments for different properties is not also an argument because it would have two maximal elements, thus violating condition 1, which requires $S$ to have a (single) least upper bound. This prevents the inclusion in the arguments of "garbage", i.e. properties not supporting the conclusion in that particular line of reasoning.

Example 42 Arguments.

\[
\begin{align*}
& b & & \begin{array}{c}
p \rightarrow r \\
p \rightarrow s \\
\end{array} \\
& a & & \begin{array}{c}
r \\
s \\
\end{array} \\
\end{align*}
\]

\[
\begin{align*}
& b-p & & b-p & & b-p \\
& a-r & & a-s & & a-r \quad a-s \\
\end{align*}
\]

arguments not argument

Figure 4.5: Multiple rules for the same property in a single class.

With respect to the HST in figure 4.5, the set of arguments for target $a$ is

\[
AR_H = \{ \begin{array}{c}
\{a-r\}, a-r, \\
\{a-s\}, a-s, \\
\{b-t\}, b-t, \\
\{a-r < b-p\}, b-p, \\
\{a-s < b-p\}, b-p \\
\end{array} \}
\]

Notice that the only argument for target $b$ is $\{b-t\}, b-t$, because the part of the hierarchy from which a certain class can inherit is restricted to itself and its own superclasses, so the facts $r$ and $s$ are not available to $b$.

The union of the two arguments for $p$ is $\{a-r, a-s < b-p\}, b-p$, but it is not an argument, because whichever rule is picked up for $p$, its body does not contain both $r$ and $s$, as it should by condition 2 in the definition of argument.
The result of adding an extra pair to the first argument \( \{a-r, a-s\}, a-r \) is also not an argument, by condition 1, because the set of pairs does not have a least upper bound.

Condition 3 has also a minimization purpose. It tackles the problem of having rules for the same property in different classes. It forbids the coexistence in an argument of pairs with the same property and different classes. Such a situation would mean the simultaneous use of different rules to get the same property, a possibility that adds nothing to the power of the system. In fact, if a rule is successfully used in one branch of the partial order, it can be also used in every other branch where the same property shows up. Combinations with other rules in the same argument become irrelevant. If the use of that rule turns out to be defeated, the whole argument is rejected, no matter which rules are used in the rest of it. Of course, the other possible rules are still influential but each in its own argument.

Example 43 A single rule for each property.

![Diagram](image)

Figure 4.6: Multiple rules for the same property in different classes.

As can be seen in figure 4.6, the poset (iii), as well as (iv), respects conditions 1 and 2 and corresponds to an actual line of reasoning. However, it implies the
existence of posets (i) and (ii), for the same least upper bound, which are smaller than (iii) and (iv), with respect to set inclusion.

The fate of these two lines of reasoning, whether accepted or rejected, will determine that of the other two and of those obtained by the union of some of them, like (v). So, only (i) and (ii) are considered arguments. □

The simultaneous use of different rules for the same property in a single argument, which was forbidden, in the case they came from the same class, by condition 2 is also ruled out, for different classes, by condition 3. In both cases prevails a minimality criterion.

Note however that minimality in the set of arguments in not fully enforced. In figure 4.7, poset (ii) contains poset (i) and both are arguments. This redundancy is present in the theory and is not introduced by our particular choice of the arguments’ structure.

![Diagram](image)

Figure 4.7: Non-minimal arguments.

A remarkable consequence of the requirement that arguments be based on irreflexive posets is the nonexistence of unfounded arguments, because no pair can belong to its own predecessors. Such unfounded lines of reasoning occur in theories containing rules like \( p \leftarrow \cdots \leftarrow p \). According to the Well-Founded Semantics [VGRS90] of Logic Programming, they are not sound, due to their self-supporting nature. The argument construction method must avoid the introduction of spurious arguments in the set \( AR_H \) because, from the Argumentation Theory point of view, all of them are equally valid and able to affect the conclusions drawn from the relation attacks.

### 4.2.2.2 Relation attacks

According to definition 119 an argument \( A \) attacks an argument \( B \) if its conclusion contradicts any of the supporting pairs of the other, provided that the former’s label class is not a superclass of the latter’s. Notice that only arguments for the same target class may interact, because different targets see different sub-hierarchies, giving rise to disjoint subsets of arguments.
It is now patent the particular kind of nonmonotonicity used by HST. Basically, the occurrence of a contradiction means that there is an attack, unless the attacked (sub)conclusion is obtained from a subclass of the agressor’s class, i.e. is in an overriding position. It is this combination of contradiction with specificity that produces a rather complex behavior of the system, in terms of defaults and overridings. Taking into consideration only the notion of contradiction and ignoring specificity, which is equivalent to collect all the theories in a single class, every attack would redound in mutual neutralization, a far less interesting situation.

The characteristic function $F_{AF}$ of an argumentation framework $AF$ is used to define the corresponding fixed-point semantics. The rationale behind it is that an argument which is not attacked must be accepted. But an attacked argument may also be accepted if the attackers are all themselves counterattacked by accepted arguments. This adds a second degree of complexity to the nonmonotonic part of the system, in addition to the already mentioned contribution of specificity.

The skeptical semantics of $AF$ is the least fixed-point of the operator $F_{AF}$. Starting with an empty set, it builds step by step the set of all accepted arguments.

**Example 44** Skeptical semantics of $AF(\mathcal{H})$.

The embedding in the argumentation framework of the HST in example 23, page 70, target $a$, is $AF(\mathcal{H}) = (AR_{\mathcal{H}}, attacks_{\mathcal{H}})$ where

$$AR_{\mathcal{H}} = \{\{a-t\}, a-t\}, \{c-r < a-p\}, a-p\}, \{b--p\}, b--p\}, \{a-t < c-s < b--r\}, b--r\}, \{b--t\}, b--t\}, \{c-r\}, c-r\}, \{a-t < c-s\}, c-s\}, \{c-r < c-u\}, c-u\},$$

and

$$attacks_{\mathcal{H}} = \{(1,5), (2,3), (4,2), (4,6), (4,8)\}.$$

![Figure 4.8: Graph of attacks.](image)
The iteration of the characteristic function reaches the least fixed point in two steps.

\[ F_{AF(\mathcal{H})}(\emptyset) = \{1, 4, 7\} \]
\[ F_{AF(\mathcal{H})}^2(\emptyset) = \{1, 3, 4, 7\} \]

In the first step, argument \(3\) cannot be accepted because it is attacked by \(2\), which on its own is not attacked by an already accepted argument. In the second step, \(4\) counter-attacks \(2\) thus defending \(3\). A third application of \(F_{AF}\) yields the same result, which is the skeptical semantics of \(AF(\mathcal{H})\), in this case coinciding with the credulous semantics. Notice that the selected arguments associate to each target class the same set of properties as the bottom-up semantics of HST did in example 23.

\[ \square \]

4.2.2.3 Correspondence

The main phenomenon at display in example 44 is the overriding. There is a direct correspondence between \(t\) overriding \(\neg t\) and \(1\) attacking \(5\) and rendering it unacceptable. In both cases, the specificity ordering decides the winner.

The other basic contradiction solving mechanism is neutralization. When a contradiction arises from two unrelated classes, like in the Nixon dilemma (see example 45), HST cancels both opponents. The embedding in the argumentation framework shows that the translation of a neutralization is a cycle with two arguments mutually attacking each other. The skeptical semantics ignores them both, too. While the overriding is an attack from below, neutralization is a side attack.

Example 45 Cascaded neutralization.

The HST in figure 4.9 is a representation of the Nixon double dilemma. Is he a pacifist? If yes, is he a draftevader? The second ambiguity depends on the first one.

The subset of arguments with target class \(n\) is:

\[ AR = \{ (\{q-p\}, q-p), \quad 1 \}
\[ (\{q-p < q-d\}, q-d), \quad 2 \}
\[ (\{r---p\}, r---p), \quad 3 \}
\[ (\{r--d\}, r--d) \} \quad 4 \}

and the relation attacks is represented in figure 4.9.
Figure 4.9: A HST and the corresponding relation *attacks*.

\[
\text{stable} \rightarrow (1, 2) \quad (1, 4) \quad (3, 4) \\
\text{unstable} \rightarrow 1 \quad 4 \quad \text{blocking} \\
\text{skeptical} \rightarrow \emptyset \quad \text{propagating}
\]

Figure 4.10: Complete extensions.

The skeptical semantics is defined by the greatest lower bound of the semilattice of complete extensions shown in figure 4.10. It is empty and thus of the ambiguity propagating type because leaving the basic contradiction ambiguous (neither 1 nor 3 belong to the grounded extension) does not impose a decision in the dependent contradiction. On the contrary, the ambiguity between 2 and 4 remains. If it were of the ambiguity blocking kind, the absence of 1 and 3 would force the presence of 4 due to the lack of support for 2. Actually, this last semantics corresponds to a complete, albeit not stable, extension. \(\square\)

A HST in which a property supports its own complement usually represents a pathological situation where a self-overriding or self-neutralizing property occurs. Example 46 shows that, in the terminology of the Argumentation Theory, the corresponding framework is controversial. Once again, the skeptical semantics agrees with HST.

**Example 46** A controversial HST.

There are just two arguments in the framework obtained from the HST in figure 4.11, for target a.

\[
AR = \{ \{b\neg p, b\neg p\}, \quad 1 \\
\{b\neg p < a\neg p, a\neg p\} \quad 2 \}
\]

The relation *attacks* contains not only the pair (2, 1) but also the self-attacking (2, 2). Any set containing 2 is not conflict-free. So, the ground
extension is empty and not stable, like in the previous example. But, while the later admitted several other complete and even stable extensions, corresponding to particular choices made between balanced arguments, connected by even cycles, the former has no stable extensions, because the odd cycle does not allow any stable choice to be done.

4.2.3 Generalizing the embedding

The next example highlights a problem that the version of embedding presented in definition 119 is not able to cope with.

Example 47 Cooperative work.

The HST in figure 4.12 has an immediate intuitive meaning. It gives \( p \) for class \( a \) as it is more specific than \( \neg p \) in every line of inheritance. However, the corresponding argumentation framework is not positive about that.

The subset of arguments with target class \( a \) is:

\[
AR = \{ \langle \{b\neg p\}, b\neg p \rangle, \langle \{c\neg p\}, c\neg p \rangle, \langle \{d\neg p\}, d\neg p \rangle, \langle \{e\neg p\}, e\neg p \rangle \}
\]
and the relation attacks is represented in figure 4.12. The grounded extension is empty. There are two stable extensions, \{1, 3\} and \{2, 4\}, of which the former agrees with \(<a>\) but the latter is completely counterintuitive.

The discrepancy is due to the incomplete modeling of the hierarchy \(H\) by definition 119. For \(AF(H)\) each argument is considered individually. Argument \(1\) attacks \(2\) without response but the attack against \(4\) is accompanied by \(4\) attacking \(1\). This is so though argument \(3\) issues an attack over \(4\), because \(1\) has no means to be aware of that. In HST both instances of \(p\) cooperate and collectively override both instances of \(\neg p\). The fact that arguments \(1\) and \(3\) are for the same conclusion \(p\) is overlooked by the argumentation theory. Notice that withdrawing \(4\) attacks \(1\) is too drastic a solution because, if (a more complex) argument \(3\) happens to be defeated then the neutralization situation would not be correctly translated in \(AF(H)\).

The next definition takes into account the cooperative overriding effect. See in figure 4.13 how example 47 becomes translated.

**Definition 120 (Embedding, ext.)** Arguments are like the arguments of definition 119, but with an extra parameter \(k\) to specify the kind. In addition to the arguments of definition 119, with \(k = m\), the following are also arguments. If, with respect to a property \(p\),

\[
(S^1, c^1-p, m), (S^2, c^2-p, m), (S^3, c^3-\neg p, m), (S^4, c^4-\neg p, m) \in AR_H
\]

such that \(c^1 < c^3, c^2 < c^4, c^1 \not< c^2, c^2 \not< c^1, c^3 \not< c^4, c^4 \not< c^3\) then

\[
(S^1, c^1-p, o), (S^2, c^2-p, o) \in AR_H.
\]

Let \(A_1 = (S, c-p, k), A_2 = (S', c'-p', k')\). Then \(A_1\) attacks \(A_2\) if and only if

1. \(k = m, k' = m\) and \(\exists c''-\neg p', c'' \not< c \in S'\); or
2. \(k = m, k' = o\) and \(\exists c''-\neg p', c'' \not< c \in S', (\neg p' \lor c < c')\); or
3. \(k = o, k' \in \{m, o\}\) and \(\neg p = p', c < c'\).

The argumentation framework for \(H\) is \(AF(H) = (AR_H, attacks_H)\).

The motivation behind the companion argument is to distribute by two arguments the overriding and neutralization forces a statement in a HST possesses. This idea could be fully explored but would result in an unnecessarily big set of arguments. In most situations a single argument can be overloaded with both roles. When the overriding is from many to many (many is more than one) then
the separation must be (partially) done in order to distinguish the nodes whose attacks are motivated by overriding. Such companion nodes cannot be attacked by neutralization. Apart from that, they suffer the same attacks as the respective main nodes. Their own attacks are also restricted to overriding, while the main nodes perform all the attacks that nodes without companion do.

Under this extended definition, the set of arguments and the graph of attacks become the ones displayed in figure 4.13. The grounded extension and unique stable extension is now \( GE = \{ 1, 3, 7, 9 \} \). Notice that the conclusions of companion arguments in \( GE \) are not in general required to belong to the semantics of HST.

\[
AR = \{ \begin{align*}
\{b-p, b-p, m\}, & \quad 1 \\
\{b-p, b-p, o\}, & \quad 1' \\
\{c-r-p, c-r-p, m\}, & \quad 2 \\
\{d-r-p, d-r-p, m\}, & \quad 3 \\
\{d-r-p, d-r-p, o\}, & \quad 3' \\
\{e-r-p, e-r-p, m\} & \quad 4
\end{align*}\]

Figure 4.13: The extended embedding.

**Example 48** The argumentation graph corresponding to the HST in figure 3.12, page 123, with respect to target a.

\[
AR = \{ \begin{align*}
\{c-q, c-q, m\}, & \quad 1 \\
\{c-q < b-p, b-p, m\}, & \quad 2 \\
\{b-s, b-s, m\}, & \quad 3 \\
\{d-r-u, d-r-u, m\}, & \quad 4 \\
\{f-r, f-r, m\}, & \quad 5 \\
\{d-r-u < f-r, f-r, m\}, & \quad 6 \\
\{b-s, (d-r-u < f-r) < c-r-p, c-r-p, m\}, & \quad 7 \\
\{a, \neg e-u, e-u, m\}, & \quad 8 \\
\{a, \neg e-u < e-r-p, e-r-p, m\} & \quad 9
\end{align*}\]

Figure 4.14: Another extended embedding.

Notice that although apparently the situation is a collective overriding, actually the rules \( p \leftarrow x \) and \( \neg p \leftarrow v \) are not mapped to arguments, due to the lack of rules for \( x \) and \( v \). So the companion arguments are not needed.

\[
F_{AF}(\emptyset) = \{ 1, 3, 4 \} \\
F_{AF}^2(\emptyset) = \{ 1, 2, 3, 4, 6 \}
\]
4.2 HST as an Argumentation Theory

\[ F^3_{AF}(\emptyset) = \{1, 2, 3, 4, 5, 6\} \]
\[ \langle a \rangle = \{q, p, s, \neg u, r\} \]

See arguments (4) and (8) to check that the rule \( \neg p \leftarrow u \) is overridden and therefore \( p \) is concluded. \( \square \)

In [Dun93a] a proposal is made for an argumentation theory with two attack relations, one stronger than the other. Equating the overriding attacks plus attacks on the support with the stronger relation and the neutralization attacks with the weaker relation leads to an argumentation framework whose grounded extension is still empty. However its single stable extension does agree with the characterization.

We will now compare the conclusions obtained via the argumentation framework with the bottom-up semantics of HST.

The supports \( S \) in the arguments contain a record of every step of the application of the common immediate consequence operator disregarding the hierarchy, i.e., without the restriction of absence of contradictory facts below the rules. Let us call \( T^*(D) \) to the fixed-point of such operator. Notice that \( T^*(D) \subseteq T^*(D) \). So to every contradiction in \( T^*(D) \) corresponds at least one attack in the argumentation framework. But there are some extra arguments which are immediately cut off by \( T^*(D) \) because they do not have any chance of becoming effective, as they are overridden by facts below.

**Lemma 121 (Non-attacked arguments)** Let \( \mathcal{H} \) be a HST. If in the argumentation framework \( AF(\mathcal{H}) \) for target a a main argument \( A = \langle S, c-p, m \rangle \) is not attacked then the property \( p \in \langle a \rangle \).

**Proof:**

By induction on the size of \( S \).

*Base case.* \( S \) is a singleton. Then \( p \) is a fact in \( D_c \). If it is not attacked then, from definition 120, either (i) there are no arguments for \( \overline{p} \) or (ii) every argument for \( \overline{p} \) used a rule in a class \( c' > c \).

In case (i), for every rule \( \overline{p} \leftarrow Q, \ Q \not\subseteq T^*(D) \), and so \( Q \not\subseteq T^*(D), \overline{p} \not\subseteq T^*(D) \). In case (ii), by (2.11), as \( p \) is a fact \( \overline{p} \not\subseteq T^*(D) \), too. So, \( p \in \langle a \rangle \), by (2.13).

*Inductive step.* If the size of \( S \) is \( k + 1 \) and \( A \) is not attacked then all the properties in \( S \), possibly except \( p \), have arguments with supports of size less or equal to \( k \), which are not attacked. Notice that if a pair \( d-q \) is in \( S \) then there is an argument with head \( d-q \) whose support is contained in \( S \).

By definition 120, an argument that attacks another on an occurrence of a pair \( b-r \) also attacks the other arguments where it occurs. The only exception to
this concerns the companion arguments. As their attacks are a subset of those performed by the respective main argument, if \( A \) is not attacked by the latter then no argument for a pair in \( S \) is attacked by the former. Thus we can ignore companion arguments.

By hypothesis, all the properties in \( S \) except \( p \) belong to \( <a> \). So, there is a rule for \( p \) in \( D_c \) with a body \( Q \subseteq <a> \). This means, by the finitude of \( Q \), that at some step \( n \) of the iteration of \( S_D \) that rule becomes a fact and, by (2.11), prevents \( \overline{p} \) from being generated in that iteration. The only rules for \( \overline{p} \) with a body in \( T^*(D) \) must belong to a \( c' > c \), given that \( A \) is not attacked. So \( p \in <a> \). –

**Theorem 122 (Soundness AF)** Let \( \mathcal{H} \) be a HST, \( AF(\mathcal{H}) \) the correspondent argumentation framework and \( a \) a target class. If a main argument belongs to the grounded extension of \( AF(\mathcal{H}) \) then its head property is in the characterization of \( a \),

\[
A = (S, c^{-p}, m) \in GE \Rightarrow p \in <a>.
\]

**Proof:**

By induction on the number of applications of the characteristic function \( F_{AF} \).

**Base case,** \( k = 1 \).

\( F_{AF} \uparrow 1 \) contains only arguments that are not attacked. By lemma 121, the respective properties belong to \( <a> \).

**Inductive step.**

\( F_{AF} \uparrow k + 1 \) contains all the arguments that are acceptable with respect to \( F_{AF} \uparrow k \). Let \( V^k = \{ A \mid A \) is attacked by \( F_{AF} \uparrow k \} \). If \( F_{AF} \uparrow k \) is conflict-free then no argument in \( V^k \) is acceptable.

Let us consider that \( A \) is an acceptable argument not in \( F_{AF} \uparrow k \). Then it is attacked only by elements of \( V^k \). But all the arguments in \( V^k \) are attacked by arguments in \( F_{AF} \uparrow k \). Let \( Z^k \) be the set of arguments in \( F_{AF} \uparrow k \) that attack attackers of \( A \).

If \( B \in Z^k \) is a main argument, by hypothesis, its head property \( r \) is in \( <a> \).

If \( B \) is a companion argument with head \( c^{-r} \), it attacks an argument \( C \in V^k \) with head \( c^{-\overline{r}}, c < c' \), which attacks \( A \) on a pair \( c''^{-r}, c' \notin c' \). If the main argument of \( B \) is in \( F_{AF} \uparrow k \) then, by hypothesis, \( r \in <a> \). If not, as the companion is in \( F_{AF} \uparrow k \), there must be an argument \( D \) with head \( c'''^{-r}, c'' \notin c \). If \( c'' \notin c''' \) then \( D \) attacks \( A \) and so \( D \in V^k \). If the same happens to all similar \( D \)'s then the main argument of \( B \) is in \( F_{AF} \uparrow k + 1 \). If \( c'' < c''' \) then there is an argument \( E \) with head \( c''^{-r} \) and it has a companion \( E^o \). If \( E^o \notin F_{AF} \uparrow k \), so that \( D \notin V^k \), then it is attacked by some argument \( G \). As the support of \( E, E^o \) is contained in the support
of $A$, every such $G$ attacks $A$. So $G \in V^k$ and $E^o, E \in F_{AF} \uparrow k + 1$. From the fact that $B \in F_{AF} \uparrow k$ and that $A$ is only attacked by $V^k$ we conclude that $r \in <a>$. 

There is an order in the iteration of $S_D$ at which $r$ is included in the safe conclusions. Let us call $n$ to the maximum of such orders for the arguments in $Z^k$. By (2.15) no property in the head of an attacker of $A$ can be generated into $T^*(D/J^n)$, using the rules implied in the support of those arguments, because at least one of the properties in the respective support is deleted for contradicting a property in $J^n$. As $A$ is attacked only by arguments in $V^k$, every other argument with a head $b \rightarrow \neg p$ contradicting property $p$ must have $b > c$ and thus is overridden in iteration $n$ where rule $p \leftarrow Q$ becomes a fact. 

To simplify notation we will write $A_p$ for an argument whose head property is $p$.

**Lemma 123 (Non-conclusion)** If, in a HST $H$, a property does not belong to the consequences of the relevant theories then, in the argumentation framework $AF(H)$, every argument for the property is attacked by an argument in the grounded extension,

$$p \notin T^*(D) \Rightarrow \forall A_p \exists B \in GE, B \text{ attacks}_H A_p.$$ 

**Proof:**

If $p \notin T^*(D)$ then, by (2.11), for every rule $p \leftarrow Q \in D_b$ either

(i) there is a fact $\neg p \leftarrow \in D_c, c < b$, or

(ii) $Q \notin T^*(D)$.

In the second case, there is a $q \in Q$ such that $q \notin T^*(D)$. The hypothesis on $p$ depends on a similar condition on $q$. According to definition 119, the arguments summarize every alternative step in this recursion, provided it is well-founded. Notice that the number of properties is finite and no two pairs in the support of an argument are allowed to repeat properties. Each argument corresponds to a finite recursion. Loops are not mapped to arguments.

Let us choose $r$ to be a property such that for every rule $r \leftarrow R$ either there is a fact $\neg r$ below it or a property in $R$ depends on $r$. Then $r$ is said to be of rank $n(r) = 0$. The rank of a property $p \notin T^*(D)$ is defined to be $n(p) = 1 + \max(n(q_i))$, where the $q_i$ are the properties in the bodies of all non-overridden rules for $p$ that are founded.

Induction on the rank of a property.

$n(r) = 0$. Every argument $A_r$ to which rule $r \leftarrow R$ is mapped is attacked by the argument $B_r$ for the respective overriding fact and not the other way round.
a) If no collective overriding of \( r \) exists then no argument attacks \( B_r \) because unfounded rules are not mapped to arguments. So, \( B_r \in GE \) and thus every \( A_r \) is attacked by \( GE \).

b) Otherwise, the companions \( B_r^c \) are not attacked. Therefore, \( B_r^c \in GE \), \( A_r \) are attacked by \( GE \) and, as the only attacks on \( B_r \) are from \( A_r \) they are accepted by \( GE \), too.

\[ n(p) = n + 1 \] Every \( A_p \) is attacked either in the head, case (i), or in the body, case (ii). In case (ii), for every \( q \in Q \) such that \( q \notin T^*(D) \) we have \( n(q) \leq n \) and by the inductive hypothesis the subargument \( A_q \) of \( A_p \) is attacked by a \( B \in GE \), which also attacks \( A_p \). In case (i), there is an argument \( A_F \) which is not attacked (or a companion argument \( A^c_p \), in the case of collective overriding) and thus belongs to \( GE \). Arguments \( A_p \) not in case (ii) are attacked by it.

Definition 124 (Victims) Let \( X \) be a set of arguments. The set of arguments attacked by \( X \) is \( V(X) = \{ A \mid B \in X, B \text{ attacks } A \} \).

If \( X \subseteq GE \) then, as \( GE \) is conflict-free, \( V(X) \cap GE = \emptyset \).

Definition 125 (Pruning) Let \( G = \langle AR_G, \text{attacks}_G \rangle \) be an argumentation framework and \( X \) a subset of its grounded extension \( G \). Then \( J = \text{prune}(G, X) \) is the argumentation framework \( \langle AR_J, \text{attacks}_J \rangle \) such that

\[
\begin{align*}
AR_J &= AR_G \setminus (X \cup V(X)) \\
\text{attacks}_J &= \text{attacks}_G \setminus \{ A \text{attacks}_G B \mid \{ A, B \} \cap (X \cup V(X)) \neq \emptyset \}.
\end{align*}
\]

Lemma 126 (Invariance) Let \( G \) be an argumentation framework, \( X \) a subset of its grounded extension \( G \) and \( J = \text{prune}(G, X) \) with grounded extension \( J \). Then

\( G = X \cup J \).

Proof:

\( G \subseteq X \cup J \). We prove by induction on the number \( k \) of iterations of the characteristic function of \( G \) that \( F^k_G(\emptyset) \subseteq X \cup J \).

Base case. Let \( A \in F_G(\emptyset) \). If \( A \in X \) the result is trivial. If \( A \not\in X \), by definition 116, \( A \) is not attacked in \( G \). So \( A \not\in V(X) \) and thus \( A \in J \).

Inductive step. Consider \( A \in F^{k+1}_G(\emptyset) \). Every \( B \) attacking \( A \) belongs to \( V(F^k_G(\emptyset)) \). If \( B \in V(X) \), by definition 125, \( B \not\in AR_J \) and so it does not attack \( A \) in \( J \). If \( B \not\in V(X) \), by hypothesis, \( B \in V(J) \) and so \( A \in J \).
4.2 HST as an Argumentation Theory

\( X \cup J \subseteq G \). The proof is now by induction on \( F_J(\theta) \): \( X \cup F_J^2(\theta) \subseteq G \). Again, if \( A \in X \subseteq G \) it is trivial.

**Base case.** \( A \in F_J(\theta) \). As it is not attacked in \( J \), every possible attack must come from \( V(X) \subseteq V(G) \) and thus \( A \in G \).

**Inductive step.** \( A \in F_J^{k+1}(\theta) \). In \( J \), it is only attacked by \( V(F_J^k(\theta)) \subseteq V(G) \), by hypothesis. In \( G \), it may be also attacked by \( V(X) \subseteq V(G) \). So \( A \in G \).

In each step of an iteration of \( F_{AF} \) it is possible to prune the graph and collect only the arguments which are not attacked. ■

**Theorem 127 (Completeness AF)** Let \( \mathcal{H} \) be a HST, a a target class and \( GE \) the grounded extension of \( AF(\mathcal{H}) \). If a property belongs to the characterization of a then there is a main argument for it in the grounded extension,

\[ p \in \langle a \rangle \Rightarrow (S, c-p, m) \in GE. \]

**Proof:**

The proof is by induction on the number \( n \) of iterations of the safe conclusions operator \( S_D \).

**Base case.** For every \( p \in J^1 = S_D(\emptyset) \) we have, by (2.13), \( p \in T^*(D) \), \( \overline{p} \not\in T^*(D) \). Let us perform a local induction on the number \( k \) of iterations of \( T_D \). If \( p \) is a fact it is mapped to an argument \( A_p \) that is attacked only by arguments for \( \overline{p} \). By lemma 123 all of them are attacked by \( GE \) so \( A_p \) is acceptable and also belongs to \( GE \). If \( p \) is a conclusion of the iteration \( k + 1 \) then all the properties in the body of the corresponding rule are mapped, by hypothesis, to arguments in \( GE \). Therefore, all attacks to the support (except the head) of \( A_p \) are defended by \( GE \). Attacks on the head, by lemma 123, are also so. Thus \( A_p \in GE \).

**Inductive step.** By hypothesis, the set of safe conclusions \( J^n \) is mapped to a set \( X \) of arguments in the grounded extension \( G \) of \( AF(\mathcal{H}) \). Let \( \mathcal{H}' \) be the HST obtained from \( \mathcal{H} \) by setting its theories \( D' = D / J^n \). We show now that \( AF(\mathcal{H}') \) is equivalent to \( \text{prune}(AF(\mathcal{H}), X) \).

By (2.15), every rule containing \( \overline{p} \), with \( p \in J^n \), is not present in \( D' \). Let \( A \) be an argument containing a pair \( c-p \) and \( B \) an element of \( X \) with head \( b-p \). If \( b \neq c \) then \( B \text{ attacks}_A \) and so \( A \) is pruned by \( X \). If \( b > c \), as \( B \in X \) and \( A \text{ attacks}_B \), there is a \( C \in X \), \( C \text{ attacks}_A \). So \( A \) does not belong to \( AR_{\mathcal{H}'} \). Rules with head \( p \) are also eliminated in \( D' \). The arguments they are mapped to belong to \( X \) and so they are also pruned.

The companion arguments attack a subset of the arguments attacked by their main arguments. Therefore they may be ignored. The only difference between
AF(\mathcal{H}') and prune(AF(\mathcal{H}), X) is that the arguments of the latter may contain some subarguments for properties \( p \) in \( J^n \). But, as the arguments with heads \( \overline{p} \) are pruned, the subarguments are not attacked. So the heads of the arguments in both grounded extensions coincide.

Step \( k \) of the iteration in \( \mathcal{H} \) is reduced to a first step in the iteration of \( \mathcal{H}' \). The proof for the base case may be iteratively reused in \( \mathcal{H}' \), until the increment \( \Delta \mathcal{P} \) becomes empty. As \( \langle a \rangle \) is \( J^n \) plus the characterization of \( a \) in \( \mathcal{H}' \) and, by lemma 126, \( G_{\mathcal{H}} = X \cup G_{\mathcal{H}'} \), the theorem is proved.

The next theorem states formally the coincidence between the characterization and the conclusions obtained from the skeptical semantics of the correspondent argumentation framework.

*Theorem 128 (Adequacy)* Let \( \mathcal{H} \) be a HST and GE the grounded extension of AF(\( \mathcal{H} \)). A property belongs to a characterization \( \langle a \rangle \) if and only if there is a main argument for it in the grounded extension,

\[ p \in \langle a \rangle \iff \langle S, c-p, m \rangle \in GE. \]

*Proof:*

The result is immediate from theorems 122 and 127.

The embedding in the argumentation theory encapsulates the forward chaining of rules and highlights the nonmonotonic behavior of HST.
Chapter 5

Related work

In this chapter we examine the relationship between HST and other approaches to inheritance reasoning, mainly inheritance networks. We show how the ambiguity blocking behavior can be approximated by HST and also what is the reason for some of the problems nets have. Several of the points discussed have already been mentioned before as inheritance networks constituted a reference since the beginning of the thesis.

5.1 HST versus inheritance networks

Inheritance networks, already presented in section 0.2, are a formalism to represent knowledge about taxonomies of properties and individuals. Exceptions and multiple inheritance are allowed.

To set a common ground for a comparison, we define a translation from nets to HST [DP92].

5.1.1 Translation

An obvious way to test the algorithms of HST (and in fact it was one of the main inspirations for this work) is applying them to known examples from the literature on inheritance networks. To do that, it is necessary to translate the inheritance net into the HST formalism.

Each node in the network is defeasible, so it must be translated into a property. But inheritance in HST is conveyed by the hierarchic relation. Therefore, information concerning the net structure must be preserved in it. Negative links block inheritance and positive links represent inheritance flow paths. So only positive links are kept. As class membership is strict, there is no correspondence between nodes and classes, except for the leaves, and new names must be created for them.
Pruning a net and its HST translation in a similar way enables a correspondence between intermediate entities. Finally, the dependency between nodes, expressed by the links, is represented by rules in the HST.

**Definition 129 (Translation)** Let $\Gamma$ be an inheritance network. A HST $H = (C, P, \prec, D)$ is its translation if and only if

$$
x \to y \in \Gamma \iff x^* \prec y^*
$$

$$
x \to y \in \Gamma \iff y \leftarrow x \in D_x^*
$$

$$
x \not\to y \in \Gamma \iff \neg y \leftarrow x \in D_x^*.
$$

A query Is $a \to p$ a conclusion of $\Gamma$? is rephrased into Is $p$ in $<a^*>$? and put to the HST obtained from $H$ by adding property $a$ to the class $a^*$. Adding property $a$ as a fact grounds the HST, otherwise unable to provide any answer, by asserting that typical $a$'s must enjoy the property of being $a$. The translation is uniform for all nodes and a single translation is enough for each network.

**Example 49 From nets to HST.**

![Diagram](image)

**Figure 5.1**: The correspondence.

The conclusions obtained from the HST, $<w^*> = \{w, m, \neg l, a\}$, coincide with those derived from the net when using the *skeptical upwards off-path* variant of inheritance reasoning. However this is not always the case, as will be seen in example 50.

A more straightforward translation could be devised, enforcing a correspondence between nodes and classes and asserting as facts in each class the ends of links starting in the corresponding node. Unfortunately, such scheme would not have generality enough to cope with dependencies between properties. In each concrete case it may be possible to find more economic representations in the HST. See, for instance, figure 2.1 on page 50.
5.1.2 Ambiguity propagating and ambiguity blocking

Among the several inference methods the most successful is the skeptical upwards off-path algorithm of [HTT90], because of its simplicity and computational properties [SL89]. More recent work on this subject showed some limitations of this approach. For instance, [MS91] criticized the inability of nets to propagate ambiguities in symmetrical nets like the double diamond. Example 50 shows how different conceptions about ambiguity lead to conclude different things from a net.

Example 50 Ambiguity blocking and propagating.

This translation refers to figure 0.2, on page 7. For that net the skeptical net reasoner would say that $n 
otightarrow a$ is a conclusion. However, considering the symmetries in the net, $n 
otightarrow a$ results more from an opportunism of the path $\rho = n \rightarrow r \rightarrow f \notightarrow a$ than from being a desirable consequence of the net. Path $\rho$ profits from the fact that its conflictor $\sigma = n \rightarrow q \rightarrow p \rightarrow a$ sees its initial segment conflicted by $\tau = n \rightarrow r \notightarrow p$. The ambiguity on $p$ kills path $\sigma$, paving the way for $\rho$ despite the apparent ambiguity on $a$. This is an example of an ambiguity blocking algorithm.

On the contrary, the kind of skepticism of HST is ambiguity propagating, as can be concluded from the translation of figure 5.2. Notice that, with respect to the class $n^*$, $S_{D^0}(\emptyset) = \{n, r, q, f\}$. Though the simplified program $D^1$ is not empty, $\Delta_{D^1} = \emptyset$ because only contradictions can be derived and so, $<n^*> = \{n, r, q, f\}$. The ambiguity on $p$ is propagated until $a$.

To get a similar effect in the net it is necessary that the zombie path [MS91] $\sigma$ be able to do his job, neutralizing $\rho$, although it is already killed by $\tau$. $\Box$

What example 50 shows is that the enforcement of a local skepticism may lead to the acceptance of conclusions which are intuitively ambiguous.

Some authors define the ideal semantics to be the intersection of all credulous extensions. It is not our belief. As in [Wag91] we think that such semantics
amounts to reasoning by cases and does not enough take into account the syntax of the representation. Floating conclusions (see figure 2.19), which happen to belong to all the (two-valued) credulous extensions, but whose derivation path contains conclusions which are not accepted, should not belong to the semantics. We prefer a directly skeptical approach, which is computationally more affordable, instead of going through the intersection. The proposed formalism for HST is in accordance with this view.

But, even without subscribing to the ideal semantics, we agree that a skeptical reasoner should not go beyond it, otherwise some conclusions with a certain amount of ambiguity are presented as sure. For this reason we criticize the ambiguity blocking reasoners and preferred the ambiguity propagating version.

In some more complex and also ambiguous nets, the ambiguity blocking algorithm produces, though for the wrong reason, some desirable conclusions which the HST formalism does not obtain. We have already seen it in examples 27 and 30, on pages 87 and 92. The first is a HST version of the floating path acceptance [MS91] problem. It studies the acceptance of a path whose sole contradictors are floating defeated paths, i.e., paths which are present in no extension but whose defeaters change from one extension to another. Such a path should belong to the skeptical semantics.

The reason for this weakness of HST is that it uses, to circumscribe the skeptical characterization, a superset of the union of the credulous characterizations. The result is that some properties which are present in no characterization have the power of neutralization in a manifestation of excessive skepticism. We still prefer our directly skeptical approach because at most it misses some intuitively acceptable conclusions but never gives the ambiguous results.

Nevertheless, this raises the question of whether there is an ambiguity blocking version of the skeptical algorithms of HST. The answer is yes and we present, without further analysis, modifications of the bottom-up operator and of the inference rules.

The idea is to extend to pairs of contradictory facts the principle that only facts are allowed to do overriding. If one can not override the other, then there is no way to avoid mutual neutralization. See in example 50 that, although it may be not clear from the HST $D^0$, it becomes evident, after the first iteration, that in the simplified HST $D^1$ the pair \( p \rightarrow \neg p \) must be neutralized. It may thus be also included in the simplification of the following iteration.

**Definition 130 (Neutralization)** The set of neutralized facts is

\[
N(D) = \{ r | r \in D_a, \overline{r} \in D_b, (\overline{\alpha} d < a, \overline{r} \leftarrow Q' \in D_d), (\overline{\beta} e < b, r \leftarrow Q'' \in D_e) \}.
\]
The set $N(D)$ contains the pairs of contradictory facts for which there are no rules with opposing heads below. This set is used to further simplify the program in conjunction with the current safe conclusions set.

**Definition 131 (Blocking)** The ambiguity blocking safe conclusions operator is

$$S'_D(J) = J \cup \Delta_D/J'$$

where $J' = J \cup N(D/J)$. The characterization of a class $a$ in a HST $(C, P^+, \prec, D)$, under ambiguity blocking rule inheritance, is then

$$<c> = S'_{(D_e)_{a \leq}} \uparrow w.$$

Like before, it is computationally more effective to incrementally simplify the program, using only the new facts. In example 50 now $\neg a \in <n^*>$ because $a$ was conditional on $p$ which is deleted together with its opposite $\neg p$ in the blocking procedure. Notice that the first neutralization step takes place before the first computation of the increment $\Delta_D$.

A similar idea can be used to modify the inference system. Derivations under the set of rules in figure 3.1 propagate ambiguities basically because the weak relation works in parallel with the strong relation and is able to produce neutralizations far away from the initial ambiguity. But if, instead of requiring that a weakly derived property be based on a weakly derived body, we substitute rules [WV], [WC], and [WR] by a new rule

$$[W] \quad \frac{\neg_a Q \quad b \overset{b}{\vdash} p}{\neg_a p} \quad \left\{ \begin{array}{l} p \leftarrow Q \in D_b \\ a \leq b \end{array} \right\}$$

a different behavior would be obtained. As now we are insisting on a strong derivation for rules’ bodies, ambiguities have only a local effect and are not propagated along subsequent rules, because a weak property cannot be used to arrive at new conclusions. It can only prevent its opposite from being strongly derived. With this modification, again in the example 50, $\neg a$ is strongly derived because there is no weak derivation for $a$ as its body $p$ is not strongly derived.

This result is closer to the skeptical upwards off-path version of the path-based reasoner in inheritance nets but it does not completely agree with it.

### 5.1.3 A deeper reason for divergence

We think that most of the success of the path-based approach derives from its naturalness and from its ability to conceal contradictions, while allowing to extract useful information from the rest of the system. What makes it strong seems, in our opinion, to be also its main weakness.
Let us review the motivating example in figure 1.1, on page 12. Assertion 4 establishes a generic dependency between flying and being light which is meaningful for animals, in general. This dependency is stated in the theory for class animal, accurately representing the expressed knowledge. It produces no result at that level, but becomes effective wherever flying is true in the corresponding subclasses and an explicit ¬light isn't in force.

In inheritance nets, adding an intermediate node between bat and mammal with a negative link to light would render the net ambiguous, because the new node cannot be considered more specific than flying on the grounds that the path bat → new → mammal ∴ flying is negative. This unfortunate result stems from the double role negative links play, by simultaneously stating an exception and blocking further reasoning. This problem becomes even clearer in the following extension to the previous example.

Example 51 Global character of links.

Add the following assertions to example 3.

7. Planes are aircrafts.
8. Aircrafts fly.
9. Planes are not light.

A representation in our proposal will have two classes, aircraft and plane as a subclass. The latter's local theory is ¬light and the former's is just flying. In terms of the extensional semantics, omitting information about weight may mean that, among the typical individuals of aircraft some are light and other ¬light.

The corresponding inheritance net also adds the two nodes, a link from aircraft to flying and a negative link between plane and light. As a result, aircraft and plane are inheriting a dependency which was stated in the completely unrelated context of animal. If plane escapes the problem by contradicting conclusion light, aircraft has no way to do the same thing while staying inside the given formulation of knowledge. Nothing is said about the weight of generic aircrafts (think of balloons, delta-wings and inter-continental jets) but it is forced to inherit the conclusion established for animal or to negate it. Eliminating the problematic link flying → light is not a solution because then every flying animal should have a second link to light therefore loosing the advantages of inheritance.

The problem in the previous example is that a link in a net has an implicit universal quantification, because any node linked to its origin will inherit the corresponding target. This again is related to the links' double role of reasoning path
and statement of properties. We think that a better result is achieved by using a different factorization of primitives, (is-a, ⟷, ¬) meaning (inheritance, dependency, negation), instead of (is-a, is-not-a). The is-a supports, at the semantic level, the taxonomic notion of inclusion among classes. The pair (⟷, ¬) is related to the notion of subset of typical individuals. There is a single notion of dependency (⟷) and overriding is obtained through contradiction (¬) and specificity, instead of being based on negative (asymmetric) links.

Example 52 Interacting defeaters.

(b)

(a)

\[
\begin{align*}
&b : \text{boring} \\
g : \text{gray} \\
e : \text{elephant} \\
r : \text{royal-elephant} \\
c : \text{clyde}
\end{align*}
\]

Figure 5.3: Are these nets ambiguous or not?

The net in figure 5.3(a) [TTH91] shows a problem with the interaction of defeaters. The path \(c \rightarrow r \rightarrow e \rightarrow g\) is preempted by \(c \rightarrow r \not\rightarrow g\). However, the link \(c \rightarrow g\), by asserting the conclusion \(g\), may reinstate the first path and thus imply that \(r\) is more specific than \(g\). If reinstatement is not allowed, i.e., if the existence of the link does not imply that the path \(c \rightarrow r \rightarrow e \rightarrow g\) be permitted, then \(r\) won't be more specific than \(g\) anymore. In fact \(r \not\rightarrow g\) is itself a link. In this case, there is an ambiguity about whether \(c\) is a \(b\) or not because \(c \rightarrow g \not\rightarrow b\) and \(c \rightarrow r \not\rightarrow b\) conflict. With reinstatement, the ambiguity is decided in favor of \(c \rightarrow r \not\rightarrow b\), by preemption, as in this case \(r\) is more specific than \(g\).

Case (b) is clearer but counterintuitive anyway. Node \(r\) is not more specific than \(g\), with respect to \(c\), because \(c \not\rightarrow e\) conflicts \(c \rightarrow r \rightarrow e\), also making \(c \rightarrow r \rightarrow e \rightarrow g\) not inheritable. But \(r \rightarrow e \rightarrow g\) is permitted ... The consequence is that the net is ambiguous on \(b\).

We argue again that this problem stems from the overloading of paths in a network with two distinct functionalities. They serve both as inheritance paths and as a means to assert properties and non-properties. Should asserting \(r \not\rightarrow g\) also forbid \(r\) from being more specific than \(g\) or not? In HST the two notions are separated making explicit the specificity which is implicit on the paths permitted by a net.
The algorithm of [HTT90] allows reinstatement so \( c \not\rightarrow b \) is a conclusion. The translation to HST gives the same result. Figure 5.4 shows two more direct representations of the knowledge contained in figure 5.3(a). In version (i) we consider that the reason for \textit{clyde} being \textit{boring} is that he is \textit{gray}. So, we state that \textit{things} are by default \(-\textit{boring}\) while \textit{elephants} are \textit{boring} if they are \textit{gray}. As \textit{royal-elephants} are \(-\textit{gray}\) the rule is not active for them and so they are \(-\textit{boring}\) by inheriting the upper property. On the contrary, \textit{clyde}, being \textit{gray} is also \textit{boring}. In version (ii), the representation takes the view that \textit{royal-elephants} are \(-\textit{boring}\) by themselves, independently of their color, and so \textit{clyde}, being one of them, inherits this attribute. The point here is that the notion of specificity is kept independent of the positive or negative properties actually inherited.

With respect to the net (b) our translation diverges from the skeptical extension. In the translated HST \( r^* < c^* < g^* \) irrespective of \( e \) being or not inherited and so \(-b\) overrides \( b\).

The hierarchy of classes with its strict inclusion acts as a backbone around which defeasible properties are grouped. This is a different view from the one expressed in [HT88] where strict and defeasible links are mixable in a more liberal way, giving rise to reasoning methods which are essentially an extension of those used with totally defeasible nets, in order to enforce backward propagation of negation along strict paths. HST is less liberal in the strict part. It does not allow a strict link after a defeasible one. But it is more versatile in the defeasible part. Dependencies among properties are not restricted to the existence of single permitted paths but include simultaneous dependencies on multiple conditions and these may be both positive and negative.
5.1.4 Expressiveness

Examining figure 5.5 it is possible to conclude the following about the expressiveness of HST:

- it allows conjunctions in the body of rules to state simultaneous dependency on several properties (e.g. dangerous ← carnivorous, large),

- reasoning may proceed after a negative property, thus allowing unbiased choice of properties (e.g., jailed ← ¬tamed),

- support for definitional statements (a lion is a feline and that cannot change, whatever new information is added), and

- it is possible to state directly dependencies among properties for a whole subsystem without asserting also the premises of the rule (there is a connection between being jailed and being ¬tamed in the context of all circus-animals although these are, by default, tamed; that rule is activated only for shaka).

The mentioned capabilities together give greater flexibility in specifying generic modules with related information at the points in the hierarchy which cover the relevant classes.

We have been essentially dealing with a propositional version of the language but the door to a lifting to first order languages is left open. For instance, shared variables in a conjunction are local to the corresponding rule. Such lifting is hard to conceive in inheritance networks because of its lack of locality.
5.2 Other logic approaches related to HST

Representing inheritance in default logic [Eth87a, Eth87b] requires semi-normal defaults to allow for exceptions (with just normal defaults the problem of interacting defaults would arise). This explicit handling of defaults has three problems: updates to the KB have no locality; complexity of individuals increases with the KB, and the translation of a link depends on the others. This representation lacks notational efficacy [Woo75] characterized by conciseness and ease of modification. Syntactically, every link in an inheritance network is a normal default. The inferential distance shows how to derive the semi-normal rules from these links. A previous representation by Etherington and Reiter [ER83] does not include the notion of overriding so it does not fully express the notion of inheritance. In chapter 3 we presented a translation of HST to Logic Programming that is modular. The translation of a rule is not affected by possible additions of other rules or classes.

Negation by failure has been used [KS90] to model defaults, through the definition of abnormality predicates. This method suffers from non-locality [Tou87], because the representation of a single assertion may involve several different rules. Like inheritance networks it also relies on a preference for negative literals, which are understood as exceptions. Although negation by failure is used for other purposes too, we restricted ourselves to the use of strong negation to concentrate on the study of inheritance of defaults along a hierarchy of theories.

Although sharing with our work a similar starting point, that of structuring a logic program as a hierarchy of modules, [McC93] follows a different approach. It is based on predicate inheritance rather than rule inheritance, although extended with a self construct, which enables the evaluation of annotated goals in a different module. As the language does not contain negative heads, exceptions must be explicitly programmed, instead of being based on a generic logic criterion using the notion of contradiction.

In [MP90, MP92] there is a comparative study of predicate inheritance, and clause inheritance in structured Logic Programming. The combination of modules is done in two different modes: extension and overriding, with respect to each predicate definition (set of clauses for the same predicate name). The two modes are needed because no explicit negation is allowed in the heads of clauses. The main difference of our approach is in the way overriding is dealt with. As HST works with positive and negative literals, it is possible to override only part of a predicate by stating just the contradictory property. It results in a finer control over the inheritance process.

The work in [Wag91] shares several intuitions with HST, for instance in the way explicit (strong) negation is viewed. Its liberal version essentially computes the same set of conclusions as the first application of the operator $T^*(D)$ except
that the latter takes preemption into account. In both approaches, this first set is used in the definition of the skeptical version. The conservative version is similar to the ambiguity blocking method for HST. In [Wag91], inheritance networks are translated into flat theories. HST uses a more uniform algorithm and keeps track of the hierarchy, resorting to it to resolve conflicts, in particular to cater for preemption. This turns out to be specially useful in a more general setting of hierarchical logic theories, which gives the programmer a richer language.

The problem of eliminating clashes by appropriately overriding defaults is central to the inheritance mechanism of HST. The solution presented relies on the hierarchy to find a criterion for that. Any property is defeasible. A similar subject has been addressed in [PAA92] but there the set of variable literals was a priori restricted to be those which were true by negation as failure. Their method of contradiction removal is based on searching for alternative sets of supports for the contradiction, which are viewed as candidates to elimination. It destroys the top-down approach because deriving a goal requires the verification that the set of found conclusions does not generate a contradiction elsewhere. Our uniform definition of neutralization, which applies even to local inconsistencies, has the advantage of assigning a semantics to every HST without the need of a priori refusing "inconsistent" programs.
Chapter 6

Conclusions

To conclude the thesis we summarize the main results of each chapter and list some open problems.

6.1 Prominent results

In conclusion, we think that the two basic intuitions that oriented this work, namely the symmetric treatment of explicit positive and negative properties and the adhesion to the LP paradigm in its constructive concern, paid off in simplicity and clarity when dealing with exceptions and incompleteness of knowledge.

The thesis clarifies the field of multiple inheritance with exceptions based on a hierarchy of theories by isolating the principles of neutralization and overriding.

The framework: syntax and extensional semantics. The adoption of a partial or 3-valued approach proved to be a natural way of formalizing incomplete knowledge about the world. The semantics obtained can deal with potential contradiction and overriding in a simple way.

The extensional semantics, being parametric on the characterization of specificity, accommodates different strategies for inheritance.

A double denotation for classes, extensional and typical, corresponding to two nested sets, is the key for interpreting defaults and exceptions.

A consistent HST entails a literal $c:p$ if and only if the property $p$ belongs to the characterization of the class $c$.

Standard models summarize the extensional semantics.

Formulas in the language can be extended to support conjunction, constructive disjunction and strong negation and the corresponding semantics is compositional.
Bottom-up construction of the characterization. Predicate inheritance in a hierarchy is intuitive because it starts the computation at the top classes and iteratively passes the respective conclusions as defaults to the subclasses. It can be computed with the help of a simple heuristic in the case of contradictions with the defaults. When contradictions arise between conclusions of a theory then it is necessary to find a minimal set of supporting defaults to override.

Predicate inheritance has a strong limitation. The functional character that a rule possesses in the class where it is defined vanishes when the class model is computed and so is not inherited by the subclasses.

Inheritance of rules cannot be satisfactorily done by successive accumulation of rules. Instead, all the relevant hierarchy is required to resolve conflicts resorting to relative specificity. The guiding line of the computation is from bodies to heads of rules and not along the hierarchy. The subordination of the hierarchy is a consequence of the restriction of the ability to override just to facts.

The intended semantics is stated via a fixed-point definition which simultaneously determine both the characterization and the weak characterization. Due to its skeptical behavior, the characterization contains the sure conclusions. The weak characterization is an upper bound on the alternative credulous characterizations that the presence of ambiguities may induce. Rule inheritance can be interpreted as the rules with heads in the characterization collectively overriding the other rules in the theories.

HST under rule inheritance enjoys hierarchic invariance in general. Property invariance is restricted to bottom classes.

This reasoner is of the skeptical ambiguity propagating kind.

In some situations, the propagation of ambiguities seems to go a little too far.

Top-down inference. The set of inference rules simultaneously define three derivation relations. The strong relation coincides with the characterization and the weak relation with the weak characterization. Neutralized properties belong the latter but not to the former. The local relation is an extension of the strong and is in charge of overriding.

Due to the negative dependencies between strong and weak relations there may be more than one triple of derivation relations induced by the rules. The least triple, with respect to set inclusion among the strong relations, is the operational semantics.

There is a modular translation from HST to Logic Programming. Each triple induced by the inference rules corresponds to a stable model of the translated
program. The least triple coincides with the well-founded model on the strong
relation. The existence of multiple stable models is related to the presence
of negative cycles in the theories.
The strong relation is a subset of the weak relation.
The inference system resorts to the notion of non-derivability. Its imple-
mentation requires a loop-detection mechanism that avoids repeated prop-
erties along a derivation path. To model the behavior of an interpreter when
negation is understood as finite failure an alternative inference system was
developed which explicitly defines the weak failure relation.

Model-theoretic views. The $\mathbb{C}$-least model of a class conveys little information
because it is always $\langle \emptyset, \mathcal{P} \rangle$. The interpretation corresponding to the Herbrand
base $\langle \mathcal{P}, \mathcal{P} \rangle$ is always a model, too. Interesting models lie in-between.
The restriction to the class of supported models in not enough to ensure good
behavior, because self-supported properties may preclude genuine conclusions
and originate inconsistency and clashes.
The stability operator settles the interpretation of negative conditions before
computing the consequences of the positive part. Its fixed-points are the
stable models, a subset of the supported models. The $\mathbb{C}$-least fixed-point is
$\langle \langle a \rangle, \langle a \rangle \rangle$, the skeptical plus the weak characterizations.
The natural embedding of HST into an argumentation framework is to equate
arguments with every possible line of reasoning, starting from the facts, and
to model the nonmonotonic interactions, i.e., neutralization and overriding,
by the relation attacks.
This solution does not completely capture the restrictions imposed by the
topology of the hierarchy on overriding. In particular, modeling the phe-
nomenon of collective overriding requires a trick involving auxiliary argu-
ments. The problem is that the argumentation framework does not provide
any form of collaboration among arguments beyond the indirect defense via
a counterattack, but with each argument individually considered.
The characterization of a HST coincides with the grounded extension of its
embedding into an argumentation framework.

Related work. A significant trend in the study of multiple inheritance with ex-
ceptions has been based on contradiction and preemption of paths in in-
heritance nets. We believe that using hierarchically structured theories with
notions of neutralization and overriding, which are directly based on negation
and specificity, is simpler and also more expressive.
The expressive advantages of HST, in comparison with inheritance networks,
include:
• dependence from multiple properties through conjunction in rules’ bodies,
• reasoning from negative properties,
• modularity in the scoping of dependencies (in inheritance networks, sub-nets above a node \( n \) are imposed to any other node linking to \( n \)), and
• ability to express definitional truths in the is-a hierarchy.

The behavior in the presence of interacting neutralizations and overridings is not always intuitive.

The semantics of HST is of the directly skeptical kind but, as it propagates the ambiguities, it does not surpass the ideal skeptical semantics, as opposed to other proposals which block ambiguity propagation.

6.2 Possible developments

Although we deliberately excluded negation as failure from the language, to concentrate on the nonmonotonic aspects of the hierarchy with strong negation, the knowledge representation task could benefit from the availability of such a default negation.

We mentioned in chapter 1 an extension of the language that allows variable arguments in properties but without functions, only constants. This version of the language is still essentially propositional but it may raise some intricate problems. If the universe of constants is not disjoint from the set \( C \) of class names, then it is possible to state in class \( nixon \), for instance, the property \( \text{likes(bird)} \). But does it imply that also \( \text{likes(tweety)} \) holds? Remember \( \text{tweety} \) is an exceptional kind of \( \text{bird} \), a \( \text{penguin} \), which does not \( \text{fly} \). And should the apparently redundant statement \( \text{is-liked-by(nixon)} \) be asserted in \( \text{bird} \)? It may be important, if exceptions are allowed on both sides of this binary relation, to represent it by the projections on each argument.

If functions are to be allowed then some of our results must be rethought, as can be concluded from example 53.

Example 53 First-order terms.

Property \( p(x) \) is considered as a short-hand for its full instantiation over the Herbrand universe \( \{a, f(a), f(f(a)), \ldots\} \). So, the three theories are infinite. In the first iteration of \( S_D \) only \( \neg p(a) \) is generated, overriding \( p(a) \). Everything else is neutralized. In the second iteration, we have also \( \neg p(f(a)) \). After an infinite number of steps, every property of the form \( \neg p(f^n(a)) \) is a safe conclusion overriding
all the $p(f^n(a))$. So, only in the step $\omega + 1$ we have $q(a) \notin T^*(D/J^\omega)$ allowing $\neg q(a)$ to become a safe conclusion.

The operator $S_D$ is not continuous in this case. As it is still monotonic, it has a fixed-point, but not necessarily in $\omega$ steps.

A last remark concerns the relationship with terminological languages. It may be profitable to search for a convergence between the ability of HST to deal with exceptions and the power of such languages to dynamically build the is-a relation.
Bibliography


[Mak88] David Makinson. General theory of cumulative inference. In M. Reinfrank, J. de Kleer, Matthew L. Ginsberg, and Erik Sandewall, editors,


Halina Przymusinska and Teodor C. Przymusinski. Semantic issues in deductive databases and logic programs. In R. B. Banerji, editor,


Appendix A

Meta-interpreter

This annex presents a simple meta-interpreter written in Prolog for HSTs represented by \texttt{isa/2} and \texttt{rule/3} unit clauses. The Prolog database must contain for each pair $a < b$ in a HST a clause $a \texttt{ isa } b$ and for each rule $p \leftarrow Q$ in $D_c$ a clause $\texttt{ rule( c, p, Q )}$.

The meta-interpreter closely follows the inference rules of figure 3.1. However, as it uses the built-in negation by failure operator not of Prolog for the negations of derivation relations, its behavior is the one implicit in the rules of figure 3.16, which makes the finite failure inference rule explicit. To compute the literals in $\texttt{\neg a}$ a loop detection mechanism must be coded that avoids both positive and negative loops. As HST is propositional, the information required by such a mechanism is simply the set of ancestors of the current goal in the computation.

\% % Meta-interpreter for Hierarchically Structured Theories %
\%
\% UNL, 93.08.20
\% Gabriel David
\%
\% \texttt{:- op(800, xfx, \_ \_ ).}
\% \texttt{:- op(800, xfx, \_ \_ ).}
\% \texttt{:- op(100, fx, \_ ).}
\% \texttt{:- op(500, xfx, isa).}
\% \texttt{:- op(500, xfx, is_a).}
\% \texttt{:- op(500, xfx, is_ax).}
\% \texttt{:- op(800, xfx, # ).}

\%
\% Strong derivation
\%
\% A \_ P - class A enjoys property P

A \_ true :- !.
A \- (P,Q) :- !, A \- P, A \- Q.

A \- P :- A is_a B, rule( B, P, Q ), compl( P, P1 ), not A \- P1, A \- Q.

%%% Weak derivation

% A \- P - there is an evidence for A to enjoy P
A \- true :- !.

A \- (P,Q) :- !, A \- P, A \- Q.

A \- P :- A is_a B, rule( B, P, Q ), compl( P, P1 ), not overel( A, B, P1 ), A \- Q.

%%% Overriding

% overel( A, B, P ) - there is a rule for P between A and B with % a strongly derived body
overel( A, B, P ) :- A is_a C, C is_ax B, rule( C, P, Q ), A \- Q.

% compl( P, Q ) - if P is atomic then Q=\-P; if P=\-R then Q=R
compl( \-P, P ) :- !.
compl( P, \-P ).

%%% Manipulation of the hierarchy

% A is_a B - reflexive transitive closure of A isa B
% A is_ax B - transitive closure of A isa B

A is_a A.
A is_a B :- A is_ax B.
A is_ax B :- A isa B.
A is_ax B :- A isa C, C is_ax B.