Defeasible Logic Programming and Belief Revision

A Tutorial for the 20th ICLP

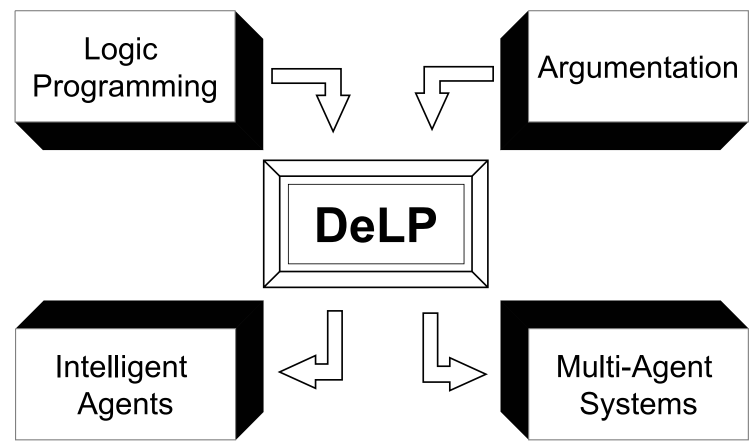
Guillermo R. Simari
Artificial Intelligence Research and Development Lab.
Dept. of Computer Science and Engineering
UNIVERSIDAD NACIONAL DEL SUR
ARGENTINA
http://cs.uns.edu.ar/~grs

Agenda

- Introduction
- Defeasible Logic Programming
- Brief Introduction to Belief Revision
- Explanations, Belief Revision and Defeasible Reasoning
- Brief List of References

Introduction

- Research in Logic Programming, Nonmonotonic Reasoning, and Argumentation has obtained important results, providing powerful tools for knowledge representation and Common Sense reasoning.
- We will introduce *Defeasible Logic Programming* (DeLP), a formalism that combines results of Logic Programming and Defeasible Argumentation.



Introduction

- DeLP adds the possibility of representing information in the form of *weak* rules in a declarative manner and a *defeasible argumentation inference mechanism* for *warranting* the conclusions that are entailed.
- Weak rules represent a key element for introducing *defeasibiliy* and they are used to represent a defeasible relationship between pieces of knowledge.
- This connection could be defeated after all things are considered.

Introduction

- General Common Sense reasoning should be defeasible in a way that is not explicitly programmed.
- Rejection should be the result of the global consideration of the corpus of knowledge that the agent performing such reasoning has at his disposal.
- Defeasible Argumentation provides a way of doing that.

Defeasible Logic Programming

DeLP's Language

- DeLP considers two kinds of program rules: *defeasible rules* to represent tentative information such as

$$\sim flies(dumbo) \prec elephant(dumbo)$$

and *strict rules* used to represent strict knowledge such as

$$mammal(idéfix) \leftarrow dog(idéfix)$$

- Syntactically, the symbol "$\prec$" is all that distinguishes a defeasible rule from a strict one.
- Pragmatically, a defeasible rule is used to represent knowledge that could be used when nothing can be posed against it.

Facts and Strict Rules

- A *Fact* is a ground literal: $innocent(joe)$
- A *Strict Rule* is denoted:

$$L_0 \leftarrow L_1,\ L_2\ ,\ \ldots,\ L_n$$

where L_0 is a ground literal called the *Head* of the rule and $L_1,\ L_2,\ \ldots,\ L_n$ are ground literals which form its *Body*.

- This kind of rule is used to represent a relation between the head and the body which is not defeasible.

Examples:

$$\sim guilty(joe) \leftarrow innocent(joe)$$
$$mammal(garfield) \leftarrow cat(garfield)$$

Defeasible Rules

- A *Defeasible Rule* is denoted:

$$L_0 \prec L_1,\ L_2,\ \ldots,\ L_n$$

where L_0 is a ground literal called the *Head* of the rule and $L_1,\ L_2,\ \ldots,\ L_n$ are ground literals which form its *Body*.

- This kind of rule is used to represent a relation between the head and the body of the rule which is tentative and its intuitive interpretation is:

"Reasons to believe in $L_1,\ L_2,\ \ldots,\ L_n$ are reasons to believe in L_0"

Examples:

$$flies(tweety) \prec bird(tweety)$$
$$\sim good\_weather(today) \prec low\_pressure(today),\ wind(south)$$

Defeasible Rules

- Defeasible rules are not default rules.
- In a default rule such as $\varphi : \psi_1, \psi_2, \ldots, \psi_n / \chi$ the justification part, $\psi_1, \psi_2, \ldots, \psi_n$, is a consistency check that contributes in the control of the applicability of this rule.
- The effect of a defeasible rule comes from a dialectical analysis made by the inference mechanism.
- Therefore, in a defeasible rule there is no need to encode any particular check, even though could be done if necessary.
- Change in the knowledge represented using DeLP's language is reflected with the sole addition of new knowledge to the representation, thus leading to better elaboration tolerance.

Defeasible Logic Program

- A *Defeasible Logic Program* (*delp*) is a set of facts, strict rules and defeasible rules denoted $\mathcal{P} = (\Pi,\ \Delta)$ where
  - Π is a set of facts and strict rules, and
  - Δ is a set of defeasible rules.
- Facts, strict, and defeasible rules are ground.
- However, we will use *"schematic rules"* containing variables.
- If R is a schematic rule, $Ground(R)$ stands for the set of all ground instances of R and

$$Ground(\mathcal{P}) = \bigcup_{R \in \mathcal{P}} Ground(R)$$

in all cases the set of individual constants in the language of $\mathcal{P}$ will be used (*see V. Lifschitz, Foundations of Logic Programming*, in *Principles of Knowledge Representation*, G. Brewka, Ed., 1996, folli)

Defeasible Logic Programming: DeLP

Here is an example of a *Defeasible Logic Program* (*delp*) denoted $\mathcal{P} = (\Pi, \Delta)$

Π

Strict Rules
$$\left\{\begin{array}{ll} bird(X) \leftarrow chicken(X) & chicken(tina) \\ bird\ (X) \leftarrow penguin(X) & penguin(opus) \\ \sim flies(X) \leftarrow penguin(X) & scared(tina) \end{array}\right\}$$
Facts

Δ

Defeasible Rules
$$\left\{\begin{array}{l} flies(X) \prec bird(X) \\ \sim flies(X) \prec chicken(X) \\ flies(X) \prec chicken\ (X),\ scared(X) \end{array}\right.$$

$$Ground(flies(X) \prec bird(X)) = \{\ flies(tina) \prec bird(tina),\ flies(opus) \prec bird(opus)\ \}$$

G.R.Simari, ICLP 2004 13

Defeasible Logic Programming: DeLP

Here is another example of a $\mathcal{P} = (\Pi, \Delta)$

Δ

Defeasible Rules
$$\left\{\begin{array}{l} has\_a\_gun(X) \prec lives\_in\_chicago(X) \\ \sim has\_a\_gun(X) \prec lives\_in\_chicago(X),\ pacifist(X) \\ pacifist(X) \prec quaker(X) \\ \sim pacifist(X) \prec republican(X) \end{array}\right.$$

Π

$$\left.\begin{array}{l} lives\_in\_chicago(nixon) \\ quaker(nixon) \\ republican(nixon) \end{array}\right\}$$
Facts

Adapted from Prakken and Vreeswijk (2000)

G.R.Simari, ICLP 2004 14

Defeasible Logic Programming: DeLP

Another example of a $\mathcal{P} = (\Pi, \Delta)$

Δ

Defeasible Rules
$$\left\{\begin{array}{l} buy\_shares(X) \prec good\_price(X) \\ \sim buy\_shares(X) \prec good\_price(X),\ risky(X) \\ risky(X) \prec in\_fusion(X,\ Y) \\ risky(X) \prec in\_debt(X) \\ \sim risky(X) \prec in\_fusion(X,\ Y),\ strong(Y) \end{array}\right.$$

Π

$$\left.\begin{array}{l} good\_price(acme) \\ in\_fusion(acme,\ estron) \\ strong(estron) \end{array}\right\}$$
Facts

G.R.Simari, ICLP 2004 15

Defeasible Derivation

Def: Let $\mathcal{P}$ = (Π, Δ) be a *delp* and L a ground literal. A *defeasible derivation* of L from $\mathcal{P}$, denoted $\mathcal{P} \mathrel{|\!\!\sim} L$, is a finite sequence of ground literals

$$L_1,\ L_2,\ \ldots,\ L_n = L,$$

such that each literal L_k in the sequence is there because:

- L_k is a fact in Π, or
- there is a rule (*strict* or *defeasible*) in $\mathcal{P}$ with head L_k and body $B_1, B_2, \ldots, B_j$, where every literal B_j in the body is some L_i appearing previously in the sequence ($i < k$).

G.R.Simari, ICLP 2004 16

Defeasible Derivation

- Notice that defeasible derivation differs from standard logical or strict derivation only in the use of defeasible, or weak, rules.
- Given a Defeasible Logic Program, a derivation for a literal L is called *defeasible* because there may exist information in contradiction with L, or the way that L is derived, that will prevent the acceptance of L as a valid conclusion.
- A few examples of defeasible derivation follow.

Defeasible Derivation

From the program:

$bird(X) \leftarrow chicken(X)$ $\quad chicken(tina)$
$bird(X) \leftarrow penguin(X)$ $\quad penguin(opus)$
$\sim flies(X) \leftarrow penguin(X)$ $\quad scared(tina)$
$flies(X) \prec bird(X)$
$\sim flies(X) \prec chicken(X)$
$flies(X) \prec chicken(X),\ scared(X)$

The following derivations could be obtained:

- $chicken(tina),\ bird(tina),\ flies(tina)$
- $chicken(tina),\ \sim flies(tina)$
- $chicken(tina),\ scared(tina),\ flies(tina)$
- $penguin(opus),\ bird(opus),\ flies(opus)$
- $penguin(opus),\ \sim flies(opus)$

Defeasible Derivation

From the program:

$buy\_shares(X) \prec good\_price(X)$
$\sim buy\_shares\ (X) \prec good\_price(X),\ risky(X)$
$risky(X) \prec in\_fusion(X,\ Y)$
$risky(X) \prec in\_debt(X)$
$\sim risky(X) \prec in\_fusion(X,\ Y),\ \ strong(Y)$
$good\_price(acme)$
$in\_fusion(acme,\ estron)$
$strong(estron)$

The following derivations could be obtained:

- $good\_price(acme),\ buy\_shares(acme)$
- $in\_fusion(acme,\ estron),\ risky(acme),\ good\_price(acme),$ $\sim buy\_shares(acme)$
- $in\_fusion(acme,\ estron),\ risky(acme)$
- $in\_fusion(acme,\ estron),\ strong(estron),\ \sim risky(acme)$

Programs and Derivations

- A program $\mathcal{P} = (\Pi,\ \Delta)$ is *contradictory* if it is possible to derive from that program a pair of complementary literals.
- Note that from the programs given as examples it is possible to derive pairs of complementary literals, such as $flies(tina),\ \sim flies(tina)$ and $flies(opus),\ \sim flies(opus)$ from the first one, and $risky(acme),\ \sim risky(acme)$ and $buy\_shares(acme),\ \sim buy\_shares(acme)$ from the second.
- Contradictory programs are useful for representing knowledge that is *potentially* contradictory.
- On the other hand, as a design restriction, the set Π should not be contradictory, because in that case the represented knowledge would be inconsistent.

Defeasible Argumentation

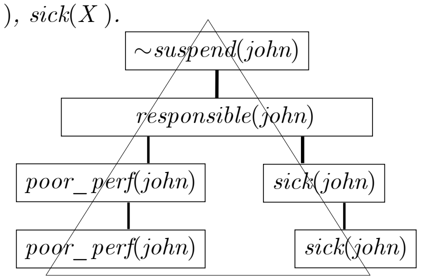
Def: Let L be a literal and $\mathcal{P} = (\Pi, \Delta)$ be a program. We say that $\mathcal{A}$ is an *argument* for L, denoted $\langle \mathcal{A}, L \rangle$, if $\mathcal{A}$ is a set of rules in Δ such that:

1) There exists a defeasible derivation of L from $\Pi \cup \mathcal{A}$; and
2) The set $\Pi \cup \mathcal{A}$ is non contradictory; and
3) There is no proper subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{A}'$ satisfies 1) and 2), that is, $\mathcal{A}$ is minimal as the defeasible part of the derivation mentioned in 1).

Defeasible Argumentation

- That is to say, an argument $\langle \mathcal{A}, L \rangle$, or an argument $\mathcal{A}$ for L, is a minimal, noncontradictory set that could be obtained from a defeasible derivation of L.
- Stricts rules are not part of the argument.
- Note that for any L which is derivable from Π alone, the empty set $\varnothing$ is an argument for L (*i.e.* $\langle \varnothing, L \rangle$).
- In this case, there is no other argument for L.

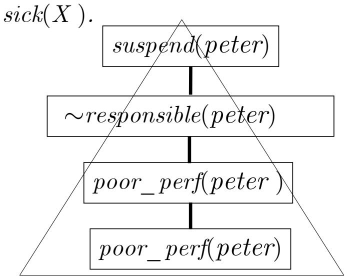
$poor\_perf(john).\ \ sick(john).$
$good\_perf(peter).\ unruly(peter).$
$suspend(X) \prec {\sim}responsible(X).$
$suspend(X) \prec unruly(X).$
${\sim}suspend(X) \prec responsible(X).$
${\sim}responsible(X) \prec poor\_perf(X).$
$responsible(X) \prec good\_perf(X).$
$responsible(X) \prec poor\_perf(X),\ sick(X).$



An argument for ${\sim}suspend(john)$ built from the program above

$\langle\{{\sim}suspend(john) \prec responsible(john).,$
$responsible(john) \prec poor\_perf(john),\ sick(john).\},\ {\sim}suspend(john)\rangle$

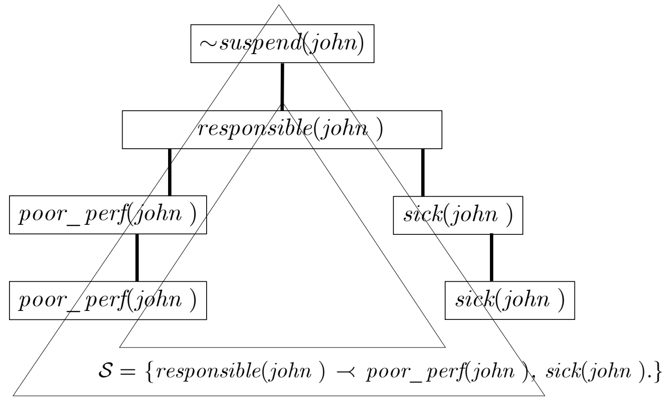
$poor\_perf(john).\ \ sick(john).$
$good\_perf(peter).\ unruly(peter)$
$suspend(X) \prec {\sim}responsible(X).$
$suspend(X) \prec unruly(X).$
${\sim}suspend(X) \prec responsible(X).$
${\sim}responsible(X) \prec poor\_perf(X).$
$responsible(X) \prec good\_perf(X).$
$responsible(X) \prec poor\_perf(X),\ sick(X).$



An argument for $suspend(peter)$ built from the program above

$\langle\{suspend(peter) \prec {\sim}responsible(peter).,$
$responsible(peter) \prec poor\_perf(peter).\},\ suspend(peter)\rangle$

$\langle \mathcal{S}, Q\rangle$ is a subargument of $\langle \mathcal{A}, L\rangle$ if $\mathcal{S}$ is an argument for Q and $\mathcal{S} \subseteq \mathcal{A}$



$$\mathcal{A} = \{\sim suspend(john) \prec responsible(john).,\ responsible(john) \prec poor\_perf(john),\ sick(john).\}$$

Rebuttal and Defeat

Rebuttals or Counter-Arguments

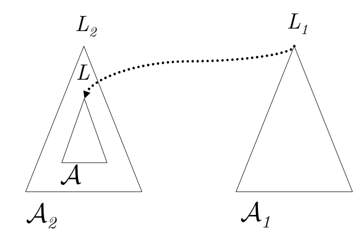
- In DeLP, answers are supported by arguments but an argument could be defeated by other arguments.
- Informally, a query L will succeed if the supporting argument for it is not defeated.
- In order to study this situation, rebuttals or counter-arguments are considered.
- Counter-arguments are also arguments, and therefore this analysis must be extended to those arguments, and so on.
- This analysis is dialectical in nature.

Rebuttals or Counter-Arguments

Def: Let $\mathcal{P} = (\Pi, \Delta)$ be a program. We will say that two literals L_1 and L_2 *disagree* if the set $\Pi \cup \{L_1, L_2\}$ is contradictory.

- For example, given $\Pi = \{ \sim L_1 \leftarrow L_2,\ L_1 \leftarrow L_3\}$ the set $\{ L_2, L_3\}$ is contradictory.

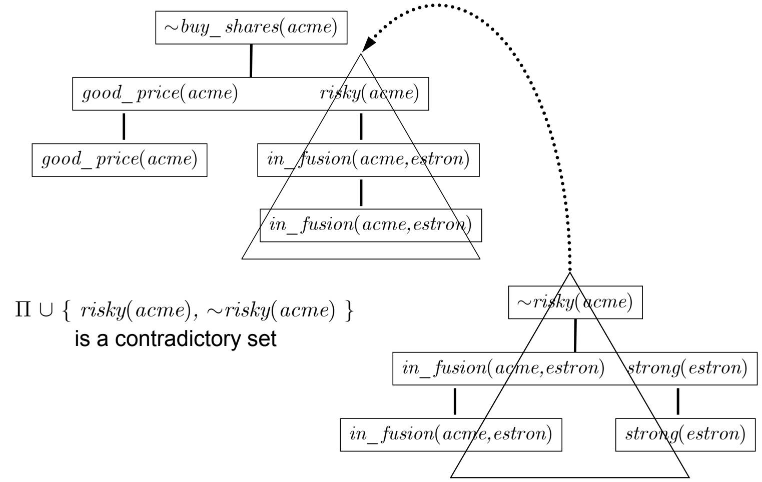
Def: Let $\mathcal{P} = (\Pi, \Delta)$ be a program. We say that $\langle \mathcal{A}_1, L_1\rangle$ *counter-argues*, *rebuts* or *attacks* $\langle \mathcal{A}_2, L_2\rangle$ at literal L, if and only if there exists a sub-argument $\langle \mathcal{A}, L\rangle$ of $\langle \mathcal{A}_2, L_2\rangle$ such that L and L_1 disagree.

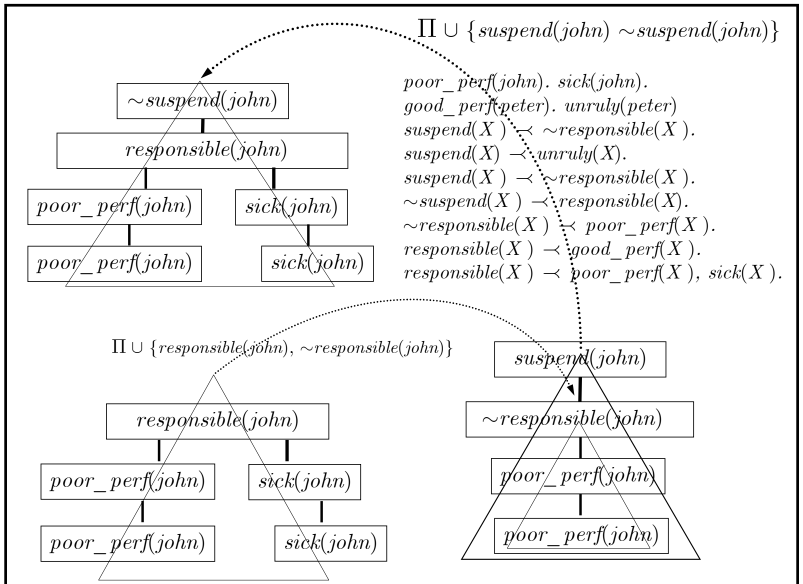


Rebuttals or Counter-Arguments

- Given $\mathcal{P} = (\Pi, \Delta)$, any literal P such that $\Pi \vdash P$, has the support of the empty argument $\langle \varnothing, P \rangle$.
- Clearly, there is no posible counter-argument for any of those P since there is no way of constructing an argument which would mention a literal in disagreement with P.
- On the other hand, any argument $\langle \varnothing, P \rangle$ cannot be a counter-argument for any argument $\langle \mathcal{A}, L \rangle$ because of the same reasons.
- It is interesting to note that given an argument $\langle \mathcal{A}, L \rangle$, that argument could contain multiple points where it could be attacked.
- Also, it would be very useful to have some preference criteria to decide between arguments in conflict.

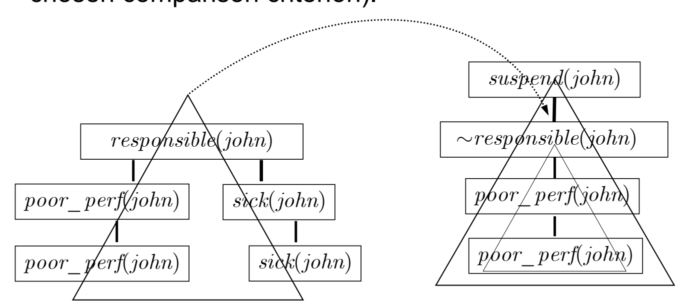
Counter-argument





Defeaters

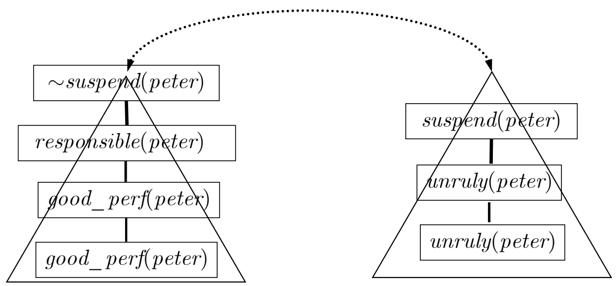
An argument $\langle \mathcal{B}, P \rangle$ is a *proper defeater* for $\langle \mathcal{A}, L \rangle$ if $\langle \mathcal{B}, P \rangle$ is a counter-argument of $\langle \mathcal{A}, L \rangle$ that attacks a subargument $\langle \mathcal{S}, Q \rangle$ of $\langle \mathcal{A}, L \rangle$ and $\langle \mathcal{B}, P \rangle$ *is better than* $\langle \mathcal{S}, Q \rangle$ (by the chosen comparison criterion).



Proper Defeater

Defeaters

An argument $\langle\mathcal{B}, P\rangle$ is a *proper defeater* for $\langle\mathcal{A}, L\rangle$ if $\langle\mathcal{B}, P\rangle$ is a counter-argument of $\langle\mathcal{A}, L\rangle$ that attacks a subargument $\langle\mathcal{S}, Q\rangle$ de $\langle\mathcal{A}, L\rangle$ and $\langle\mathcal{B}, P\rangle$ *is not comparable to* $\langle\mathcal{S}, Q\rangle$ (by the chosen comparison criterion)



Blocking Defeater

Argument Comparison: Generalized Specificity

Def: Let $\mathcal{P} = (\Pi, \Delta)$ be a program. Let Π_G be the set of strict rules in Π and let $\mathcal{F}$ be the set of all literals that can be defeasibly derived from $\mathcal{P}$. Let $\langle\mathcal{A}_1, L_1\rangle$ and $\langle\mathcal{A}_2, L_2\rangle$ be two arguments built from $\mathcal{P}$, where $L_1, L_2 \in \mathcal{F}$.
Then $\langle\mathcal{A}_1, L_1\rangle$ is *strictly more specific than* $\langle\mathcal{A}_2, L_2\rangle$ if:

1. For all $\mathcal{H} \subseteq \mathcal{F}$, if there exists a defeasible derivation $\Pi_G \cup \mathcal{H} \cup \mathcal{A}_1 \mathrel{|\!\sim} L_1$ while $\Pi_G \cup \mathcal{H} \not\mathrel{|\!\sim} L_1$ then $\Pi_G \cup \mathcal{H} \cup \mathcal{A}_1 \mathrel{|\!\sim} L_2$, and
2. There exists $\mathcal{H}' \subseteq \mathcal{F}$ such that there exists a defeasible derivation $\Pi_G \cup \mathcal{H}' \cup \mathcal{A}_2 \mathrel{|\!\sim} L_2$ and $\Pi_G \cup \mathcal{H}' \not\mathrel{|\!\sim} L_2$ but $\Pi_G \cup \mathcal{H}' \cup \mathcal{A}_1 \not\mathrel{|\!\sim} L_1$

(Poole, David L. (1985). *On the Comparison of Theories: Preferring the Most Specific Explanation.* pages 144—147 Proceedings of 9th IJCAI.)

Argument Comparison: Generalized Specificity

- Intuitively, this criteria prefers arguments with greater informational content (*i.e. more precise*) and with less use of rules (*i.e. more concise*).
- For example, from program:

$bird(X) \leftarrow chicken(X)$	$chicken(tina)$
$flies(X) \prec bird(X)$	$scared(tina)$
$\sim flies(X) \prec chicken(X)$	
$flies(X) \prec chicken(X), scared(X)$	

It is possible to obtain

$\langle\mathcal{A}_1, \sim flies(tina)\rangle$ with $\mathcal{A}_1 = \{\sim flies(tina) \prec chicken(tina)\ \}$

$\langle\mathcal{A}_2, flies(tina)\rangle$ with $\mathcal{A}_2 = \{\ files(tina) \prec bird(tina)\ \}$

$\langle\mathcal{A}_3, flies(tina)\rangle$ with $\mathcal{A}_3 = \{\ flies(tina) \prec chicken(tina), scared(tina)\ \}$

- $\mathcal{A}_3$ is preferred to $\mathcal{A}_1$ because it is more precise more information).
- $\mathcal{A}_1$ is preferred to $\mathcal{A}_2$ because it is more concise (direct).

Argument Comparison: Rule's Priorities

Def: Let $\mathcal{P} = (\Pi, \Delta)$ be a program, and let "$>$" be a partial order defined on the defeasible rules in Δ.
Let $\langle\mathcal{A}_1, L_1\rangle$ and $\langle\mathcal{A}_2, L_2\rangle$ be two arguments obtained from $\mathcal{P}$. We will say that $\langle\mathcal{A}_1, L_1\rangle$ *is preferred to* $\langle\mathcal{A}_2, L_2\rangle$ if the following conditions are verified:

1. If there exists at least a rule $r_a \in \mathcal{A}_1$ and a rule $r_b \in \mathcal{A}_2$ such that $r_a > r_b$; and
2. There is no pair of rules $r'_a \in \mathcal{A}_1$ and $r'_b \in \mathcal{A}_2$ such that $r'_b > r_a$

Argument Comparison: Rule's Priorities

From the program:

$buy\_shares(X) \prec good\_price(X)$	$good\_price(acme)$
$\sim buy\_shares(X) \prec risky(X)$	$in\_fusion(acme,\ estron)$
$risky(X) \prec in\_fusion(X,\ Y)$	

with rule preference:

$\sim buy\_shares(X) \prec risky(X) > buy\_shares(X) \prec good\_price(X)$

argument $\langle \mathcal{A}, \sim buy\_shares(acme) \rangle$ where

$\mathcal{A} = \{\sim buy\_shares(acme) \prec risky(acme),$
$risky(acme) \prec in\_fusion(acme,\ estron)\}$

will be preferred to argument

$\langle \mathcal{B},\ buy\_shares(acme) \rangle$ where

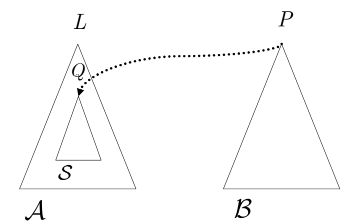
$\mathcal{B} = \{buy\_shares(acme) \prec good\_price(acme)\ \}$

Defeaters

An argument $\langle \mathcal{B},\ P \rangle$ is a *defeater* for $\langle \mathcal{A},\ L \rangle$ if $\langle \mathcal{B},\ P \rangle$ is a counter-argument for $\langle \mathcal{A},\ L \rangle$ that attacks a subargument $\langle \mathcal{S},\ Q \rangle$ de $\langle \mathcal{A},\ L \rangle$ and one of the following conditions holds:

(a) $\langle \mathcal{B},\ P \rangle$ *is better than* $\langle \mathcal{S},\ Q \rangle$ (*proper defeater*), or

(b) $\langle \mathcal{B},\ P \rangle$ *is not comparable to* $\langle \mathcal{S},\ Q \rangle$ (*blocking defeater*)



Defeaters: Example

From the program:

$buy\_shares(X) \prec good\_price(X)$	$good\_price(acme)$
$\sim buy\_shares(X) \prec risky(X)$	$in\_fusion(acme,\ estron)$
$risky(X) \prec in\_fusion(X,\ Y)$	

With preference:

$\sim buy\_shares(X) \prec risky(X) > buy\_shares(X) \prec good\_price(X)$

The argument $\langle \mathcal{A}, \sim buy\_shares(acme) \rangle$ where

$\mathcal{A} = \{\sim buy\_shares(acme) \prec risky(acme),$
$risky(acme) \prec in\_fusion(acme,\ estron)\}$

is counter-argument of

$\langle \mathcal{B},\ buy\_shares(acme) \rangle$

where $\mathcal{B} = \{\ buy\_shares(acme) \prec good\_price(acme)\ \}$

that is a proper defeater of it.

Defeaters: Example

From the program:

$pacifist(X) \prec quaker(X)$

$\sim pacifist(X) \prec republican(X)$

$quaker(nixon)$

$republican(nixon)$

With the preference defined by specificity:

$\langle \mathcal{A}, \sim pacifist(nixon)\ \rangle$ where

$\mathcal{A} = \{\sim pacifist(nixon) \prec republican(nixon)\ \}$

it is a blocking defeater for

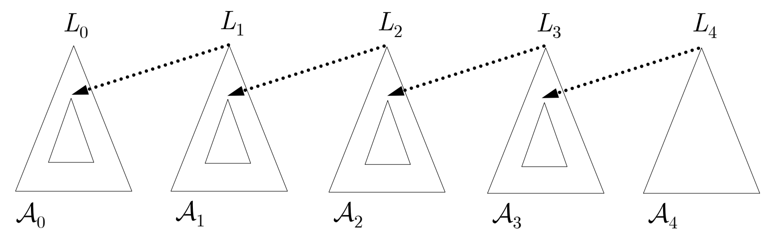
$\langle\ \mathcal{B},\ pacifist(nixon)\ \rangle$

where $\mathcal{B} = \{\ pacifist(nixon) \prec quaker(nixon)\ \}$

Argumentation Lines

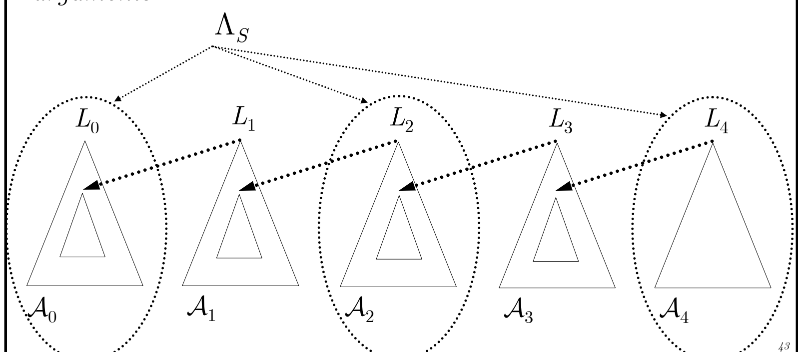
Argumentation Line

Given $\mathcal{P} = (\Pi, \Delta)$, and $\langle \mathcal{A}_0, L_0 \rangle$ an argument obtained from $\mathcal{P}$. An *argumentation line* for $\langle \mathcal{A}_0, L_0 \rangle$ is a sequence of arguments obtained from $\mathcal{P}$, denoted $\Lambda = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots]$ where each element in the sequence $\langle \mathcal{A}_i, L_i \rangle$, $i > 0$ is a defeater for $\langle \mathcal{A}_{i\text{-}1}, L_{i\text{-}1} \rangle$.



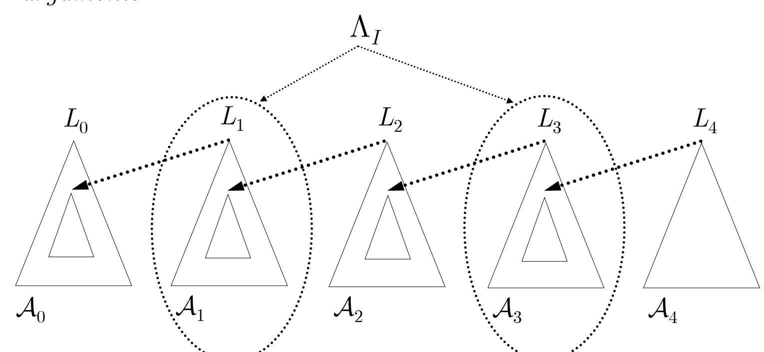
Argumentation Line

Given an argumentation line $\Lambda = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots]$, the subsequence $\Lambda_S = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_2, L_2 \rangle, \ldots]$ contains *supporting arguments* and $\Lambda_I = [\langle \mathcal{A}_1, L_1 \rangle, \langle \mathcal{A}_3, L_3 \rangle, \ldots]$ are *interfering arguments*.



Argumentation Line

Given an argumentation line $\Lambda = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots]$, the subsequence $\Lambda_S = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_2, L_2 \rangle, \ldots]$ contains *supporting arguments* and $\Lambda_I = [\langle \mathcal{A}_1, L_1 \rangle, \langle \mathcal{A}_3, L_3 \rangle, \ldots]$ are *interfering arguments*.



Argumentation Lines

Let's consider a program $\mathcal{P}$ where:

$\langle\mathcal{A}_1, L_1\rangle$ defeats $\langle\mathcal{A}_0, L_0\rangle$
$\langle\mathcal{A}_2, L_2\rangle$ defeats $\langle\mathcal{A}_0, L_0\rangle$
$\langle\mathcal{A}_3, L_3\rangle$ defeats $\langle\mathcal{A}_1, L_1\rangle$
$\langle\mathcal{A}_4, L_5\rangle$ defeats $\langle\mathcal{A}_2, L_2\rangle$
$\langle\mathcal{A}_5, L_5\rangle$ defeats $\langle\mathcal{A}_2, L_2\rangle$

Then, from $\langle\mathcal{A}_0, L_0\rangle$ there exist several argumentation lines such as:

$$\Lambda_1 = [\langle\mathcal{A}_0, L_0\rangle, \langle\mathcal{A}_1, L_1\rangle, \langle\mathcal{A}_3, L_3\rangle]$$
$$\Lambda_2 = [\langle\mathcal{A}_0, L_0\rangle, \langle\mathcal{A}_2, L_2\rangle, \langle\mathcal{A}_4, L_4\rangle]$$
$$\Lambda_3 = [\langle\mathcal{A}_0, L_0\rangle, \langle\mathcal{A}_2, L_2\rangle, \langle\mathcal{A}_5, L_5\rangle]$$

G.R.Simari, ICLP 2004 45

Argumentation Lines: Problems

- There are several undesired situations that could appear in argumentation lines.
- Let's see an example:

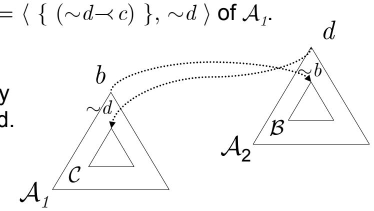
$\{ (d \prec \sim b,\ c),\ (b \prec \sim d,\ a),\ (\sim b \prec a),\ (\sim d \prec c),\ (a),\ (c) \}$

$\langle\ \mathcal{A}_1,\ b\ \rangle = \langle\ \{\ (b \prec\ \sim d,\ a),\ (\sim d \prec c)\ \},\ b\ \rangle$ is a proper defeater of

$\langle\ \mathcal{A}_2,\ d\ \rangle = \langle\ \{\ (d \prec\ \sim b,\ c),\ (\sim b \prec a)\ \},\ d\ \rangle$ and reciprocally.

Note $\langle\mathcal{A}_1,\ b\rangle$ is strictly more specific than the sub-argument $\langle\mathcal{B},\ \sim b\ \rangle = \langle\ \{\ (\sim b \prec a)\ \},\ \sim b\ \rangle$ of $\mathcal{A}_2$ and $\langle\mathcal{A}_2,\ d\ \rangle$ is strictly more specific than the sub-argument $\langle\ \mathcal{C},\ \sim d\ \rangle = \langle\ \{\ (\sim d \prec c)\ \},\ \sim d\ \rangle$ of $\mathcal{A}_1$.

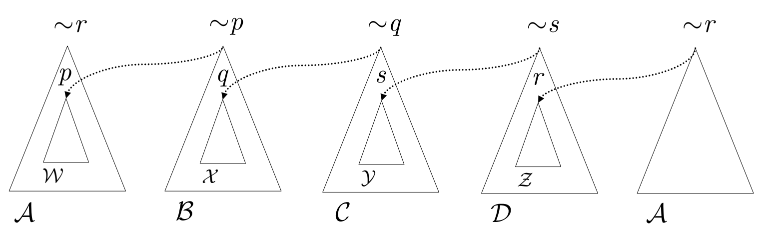
This will not be allowed since only *defeaters* could be introduced.



G.R.Simari, ICLP 2004 46

Argumentation Lines: Problems

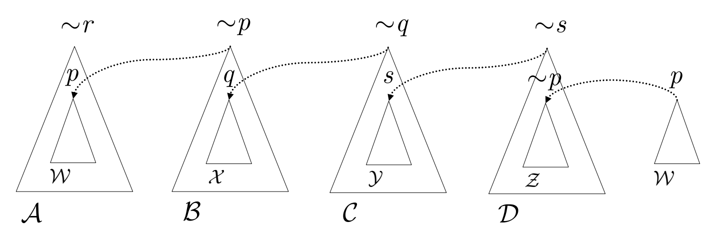
- The figure below shows another possible problem, this leading to an infinite argumentation line.
- In this case, the same argument is introduced again in the same role that was introduced before (supporting).
- The obvious solution is not to allow that.



G.R.Simari, ICLP 2004 47

Argumentation Lines: Problems

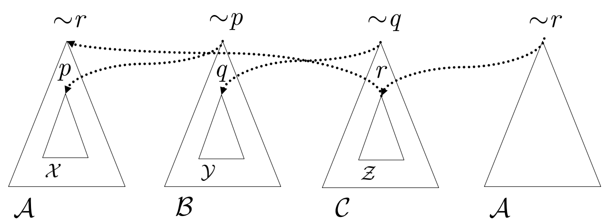
- Nevertheless, in a more subtle way, it is possible to introduce a sub-argument of an argument that is already introduced.
- When $\langle\mathcal{W},\ p\rangle$ is introduced, that action allows to reintroduce $\langle\mathcal{B},\ \sim p\rangle$ and that leads to circular argumentation.
- The problem came from the introduction of argument $\langle\mathcal{W},\ p\rangle$.



G.R.Simari, ICLP 2004 48

Argumentation Lines: Problems

- In the picture below, the argumentation line shows the problem created by reintroducing an argument.
- This argument started as a supporting argument and it is reintroduced as an interference argument.
- The problem appears when argument $\langle \mathcal{C}, \sim q \rangle$ is introduced as a supporting argument, but it contains a counter-argument for the original argument.



Flipping sides

Argumentation Lines: Problems

- This leads to the notion of *concordance* in a line.
- Given a program $\mathcal{P} = (\Pi, \Delta)$, we will say that $\langle \mathcal{A}_1, L_1 \rangle$ is concordant with $\langle \mathcal{A}_2, L_2 \rangle$ if and only if $\Pi \cup \mathcal{A}_1 \cup \mathcal{A}_2$ is non contradictory.
- In general, a set of arguments $\{ \langle \mathcal{A}_i, L_i \rangle, i{=}1,\ldots,n \}$ is said to be concordant if:

$$\Pi \bigcup \cup_{i=1}^{n} \mathcal{A}_i$$

  is non-contradictory.
- We will require that in an argumentation line the set of supporting arguments be concordant and the set of interfering arguments be concordant.

Argumentation Lines: Problems

Let's see another problem through the following example:

$dangerous(X) \prec tiger(X)$	$tiger(hobbes)$
$\sim dangerous(X) \prec baby(X)$	$baby(hobbes)$
$\sim dangerous(X) \prec pet(X)$	$pet(hobbes)$

with preference defined by specificity:

$\langle \mathcal{A}_1, \sim dangerous(hobbes) \rangle$ where $\mathcal{A}_1 = \{ \sim dangerous(hobbes) \prec baby(hobbes) \}$

will be blocked by

$\langle \mathcal{A}_2, dangerous(hobbes) \rangle$ where $\mathcal{A}_2 = \{ dangerous(hobbes) \prec tiger(hobbes) \}$

which in turn will be blocked by

$\langle \mathcal{A}_3, \sim dangerous(hobbes) \rangle$ where $\mathcal{A}_3 = \{ \sim dangerous(hobbes) \prec pet(hobbes) \}$

the line $[\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3]$ could be obtained but that will be incorrect since $\mathcal{A}_2$ was already blocked by $\mathcal{A}_1$ and that would represent the policy that having two arguments blocking a third is better than using only one argument to do that.

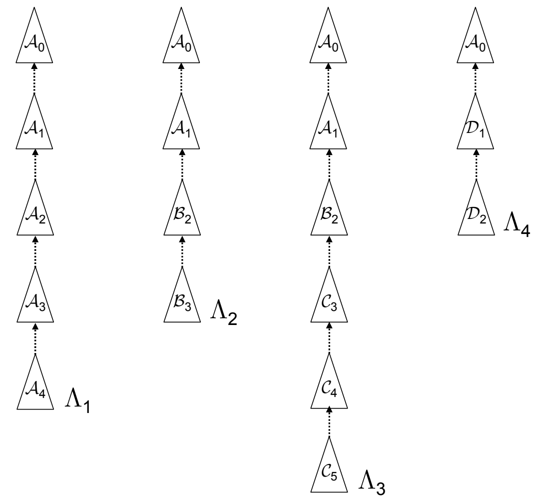
Acceptable Argumentation Line

Given a program $\mathcal{P} = (\Pi, \Delta)$, an argumentation line $\Lambda = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots]$ will be *acceptable* if:

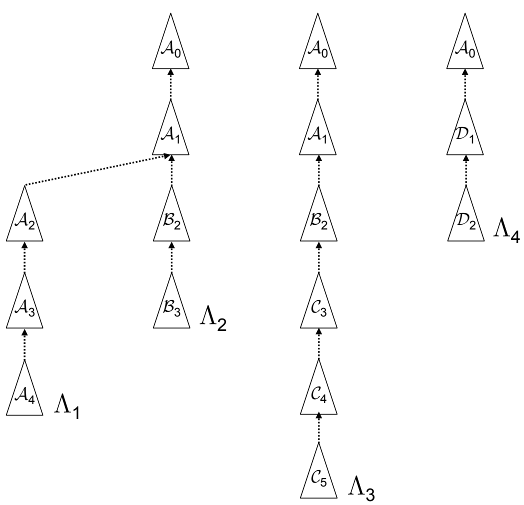
1. Λ is a finite sequence (no circularity).
2. The set Λ_S, of supporting arguments is concordant, and the set Λ_I, of interfering arguments is concordant.
3. There is no argument $\langle \mathcal{A}_k, L_k \rangle$ in Λ that is a subargument of a preceeding argument $\langle \mathcal{A}_i, L_i \rangle$, $i < k$.
4. For all i, such that $\langle \mathcal{A}_i, L_i \rangle$ is a blocking defeater for $\langle \mathcal{A}_{i-1}, L_{i-1} \rangle$, if there exists $\langle \mathcal{A}_{i+1}, L_{i+1} \rangle$ then $\langle \mathcal{A}_{i+1}, L_{i+1} \rangle$ is a proper defeater for $\langle \mathcal{A}, L_i \rangle$ (*i.e.,* $\langle \mathcal{A}, L_i \rangle$ could not be blocked).

Argumentation Lines

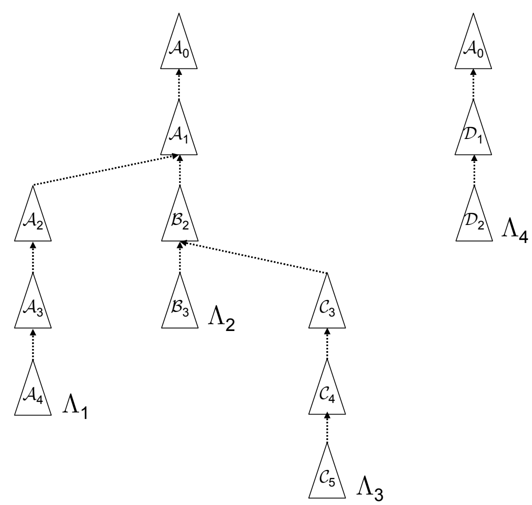
Argumentation Lines



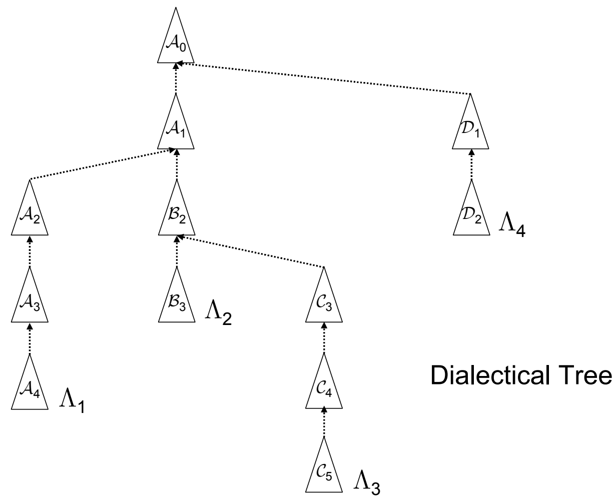
Argumentation Lines



Argumentation Lines



Argumentation Lines



Dialectical Tree

- A Dialectical Tree is the conjoint representation of all the acceptable argumentation lines.
- Given an argument $\mathcal{A}$ for a literal L, the dialectical tree contains *all* acceptable argumentation lines that start with that argument.
- In that manner, the analysis of the defeat status for a given argument could be carried out on the dialectical tree.
- As every argumentation line is admisible, and therefore finite, every dialectical tree is also finite.

Dialectical Tree

Def: Let $\langle \mathcal{A}_0, L_0 \rangle$ be an argument built from a program $\mathcal{P} = (\Pi, \Delta)$.

A dialectical tree for $\langle \mathcal{A}_0, L_0 \rangle$, denoted $\mathcal{T}_{\langle \mathcal{A}_0, L_0 \rangle}$ is defined as follows:

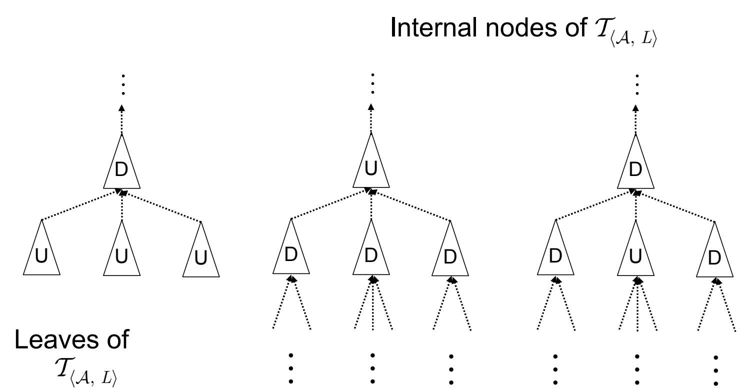
1. The root of the tree is labeled $\langle \mathcal{A}_0, L_0 \rangle$
2. Let N be non-root node of the tree labeled $\langle \mathcal{A}_n, L_n \rangle$, and

   $\Lambda = [\ \langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots, \langle \mathcal{A}_n, L_n \rangle\ ]$ the sequence of labels of the path from the root to N. Let $\langle \mathcal{B}_1, Q_1 \rangle, \langle \mathcal{B}_2, Q_2 \rangle, \ldots, \langle \mathcal{B}_k, Q_k \rangle$ be all the defeaters for $\langle \mathcal{A}_n, L_n \rangle$.

   For each defeater $\langle \mathcal{B}_i, Q_i \rangle$ $(1 \leq i \leq k)$, such that the argumentation line $\Lambda' = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots, \langle \mathcal{A}_n, L_n \rangle, \langle \mathcal{B}_i, Q_i \rangle]$ is acceptable, then the node N has a child N_i labeled $\langle \mathcal{B}_i, Q_i \rangle$.

   If there is no defeater for $\langle \mathcal{A}_n, L_n \rangle$ or there is no $\langle \mathcal{B}_i, Q_i \rangle$ such that Λ' is acceptable, then N is a leaf.

Marking of a Dialectical Tree

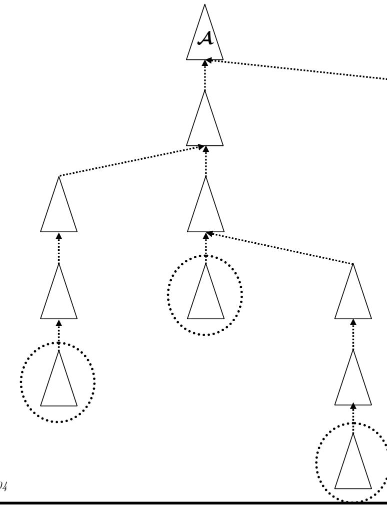


Marking of a Dialectical Tree

Marking Procedure: Let $\mathcal{T}_{\langle \mathcal{A},\, L\rangle}$ be a dialectical tree for $\langle \mathcal{A},\, L\rangle$. The corresponding marked dialectical tree, $\mathcal{T}^*_{\langle \mathcal{A},\, L\rangle}$, will be obtained marking every node in $\mathcal{T}_{\langle \mathcal{A},\, L\rangle}$ as follows:

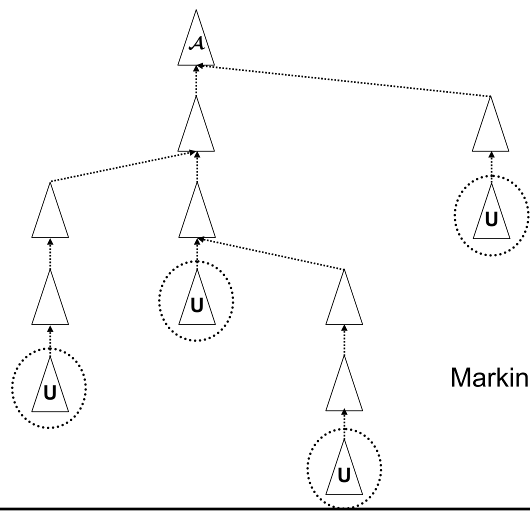
1. All leaves in $\mathcal{T}_{\langle \mathcal{A},\, L\rangle}$ are marked as U's in $\mathcal{T}^*_{\langle \mathcal{A},\, L\rangle}$.
2. Let $\langle \mathcal{B},\, Q\rangle$ be an inner node of $\mathcal{T}_{\langle \mathcal{A},\, L\rangle}$. Then $\langle \mathcal{B},\, Q\rangle$ will be marked as U in $\mathcal{T}^*_{\langle \mathcal{A},\, L\rangle}$ if and only if every child of $\langle \mathcal{B},\, Q\rangle$ is marked as D and the node $\langle \mathcal{B},\, Q\rangle$ will be marked as D if and only if it has at least a child marked as U.

Dialectical Tree



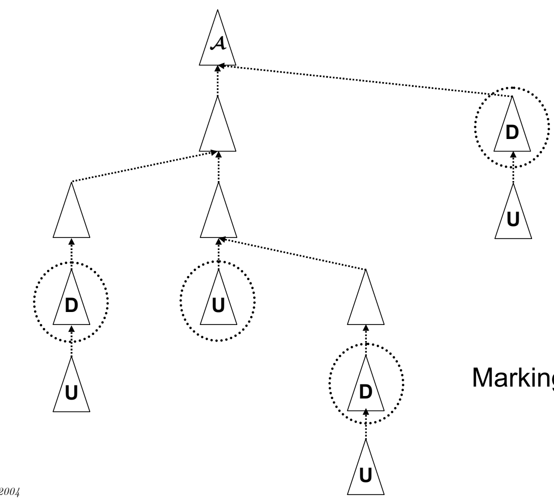
Marking

Dialectical Tree



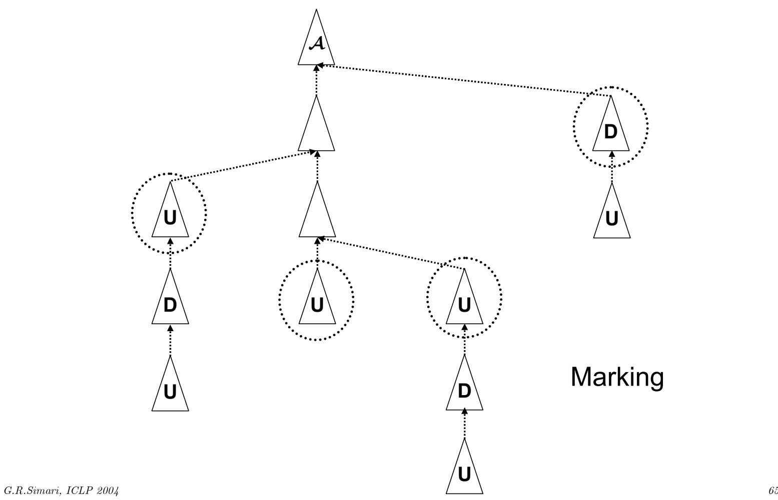
Marking

Dialectical Tree

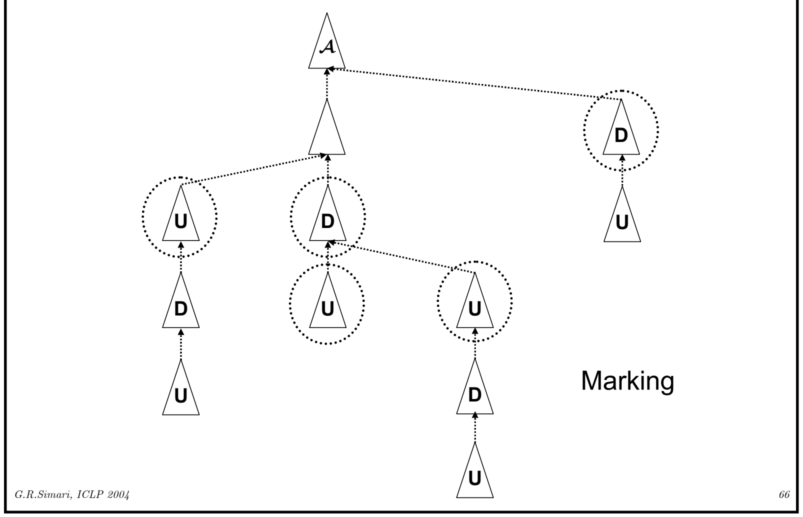


Marking

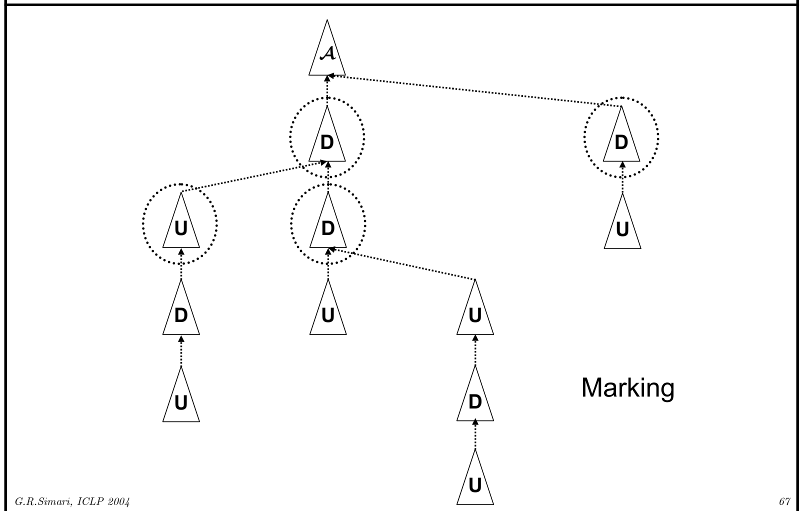
Dialectical Tree



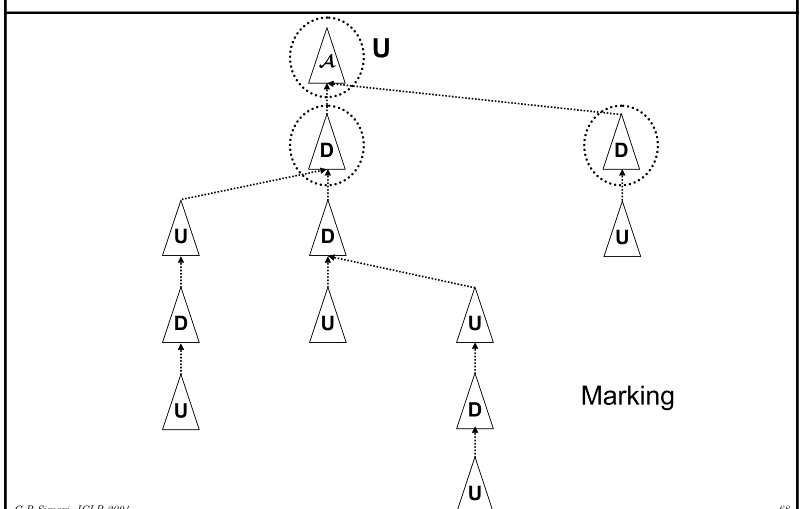
Dialectical Tree



Dialectical Tree



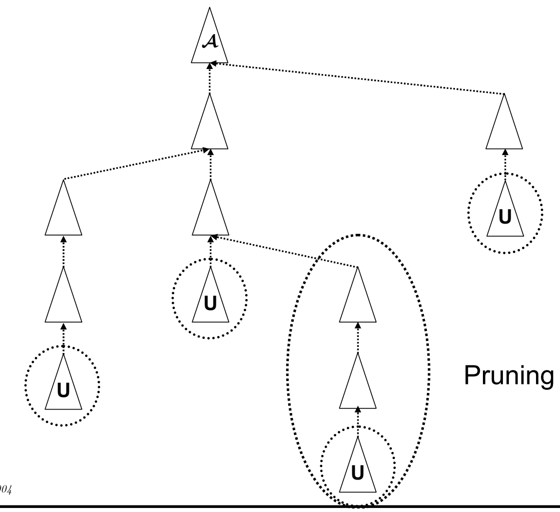
Dialectical Tree



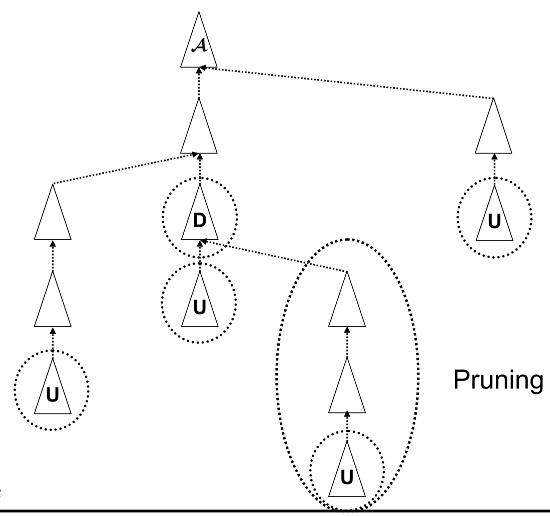
Warranted Literals

- Let $\mathcal{P} = (\Pi, \Delta)$ be a defeasible program. Let $\langle \mathcal{A}, L \rangle$ be an argument and let $\mathcal{T}^*_{\langle \mathcal{A}, L \rangle}$ be its associated dialectical tree. A literal L is *warranted* if and only if the root of $\mathcal{T}^*_{\langle \mathcal{A}, L \rangle}$ is marked as “U”.
- That is, the argument $\langle \mathcal{A}, L \rangle$ is an argument such that each possible defeater for it has been defeated.
- We will say that $\mathcal{A}$ *is a warrant for* L.

Dialectical Tree: Pruning



Dialectical : Pruning Tree



Answers in DeLP

- If the strict part Π of a program $\mathcal{P} = (\Pi, \Delta)$ is inconsistent, any literal can be derived.
- When it is possible to defeasible derive a pair of complementary literals $\{ L, \sim L \}$ it is possible to introduce a way to try to decide whether to accept one of them.
- Therefore, there are three different possible answers: accept L, accept $\sim L$, or to reject both.
- Also, if the program is used as a device to resolve queries, a fourth possibility appears: the literal for which the query is made is unknown to the program.

Answers in DeLP

Given a program $\mathcal{P} = (\Pi, \Delta)$, and a query for L the posible answers are:

- *YES*, if L is warranted.
- *NO*, if $\sim L$ is warranted.
- *UNDECIDED*, if neither L nor $\sim L$ are warranted.
- *UNKNOWN*, if L is not in the language of the program.

Specification of the Warrant Procedure

```
warrant(Q, A) :-                              % Q is a warranted literal
      find_argument(Q, A),                    % if A is an argument for Q
      \+ defeated(A, [support(A, Q)]).        % and A is not defeated

defeated(A, ArgLine) :-                       % A is defeated
      find_defeater(A, D, ArgLine),           % if there is a defeater D for A
      acceptable(D, ArgLine, NewLine),        % acceptable within the line
      \+ defeated(D, NewLine).                % and D is not defeated

find_defeater(A, D) :-                        % C is a defeater for A
      find_counterarg(A, D, SubA),            % if C counterargues A in SubA
      \+ better(SubA, D).                     % and SubA is not better than C
```


Extensions and Applications

Adding not

- DeLP program rules can contain not as in

  $\sim cross\_railway\_tracks \prec not\ \sim train\_is\_coming$

  $\sim cross\_railway\_tracks \prec cannot\_wait,$
  $not\ \sim train\_is\_coming$

- Is very simple to extend the notions of defeasible derivation, argument and counter-argument.
- If $not\ L$ is a literal used in the body of a rule, there is a new kind of attack on it, *i.e.* if we have an undefeated argument for L then the argument that contains a rule with $not\ L$ will be defeated.

Work in Progress

- Extending generalized specificity allowing utility values for facts and rules, giving the possibility of introducing pragmatic considerations.
- Decision-Theoretic Defeasible Logic Programming will be represented as $\mathcal{P} = (\Pi, \Delta, \Phi, \mathbf{B})$, where Π and Δ are as before, $\mathbf{B}$ is a Boolean algebra with top $\top$ and bottom $\perp$, and Φ is defined Φ: $\Pi \cup \Delta \to \mathbf{B}$.
- Paper in the 2004 Non Monotonic Reasoning Conf. *http://www.pims.math.ca/science/2004/NMR/add.html* or *http://cs.uns.edu.ar/~grs*

Work in Progress

- We just got the second place in the Robocup e-league using Prolog (see *http://cs.uns.edu.ar/~gis/robocup-TDP.htm.*)
  Now we are extending DeLP in a way of controlling the robots,
- An action A will be an ordered triple ⟨X, P, C⟩, where X is a consistent set of literals representing consequences of executing A, P is a set of literals representing preconditions for A, C is a set of constrains of the form $not\ L$, where L is a literal.
- Actions will be denoted:

$$\{X_1, \ldots, X_n\} \xleftarrow{A} \{P_1, \ldots, P_m\},\ not\,\{C_1, \ldots, C_k\}$$

where $not\,\{C_1, \ldots, C_k\}$ means $\{not\ C_1, \ldots, not\ C_k\}$ and $not\ C_i$ means C_i is not warranted.

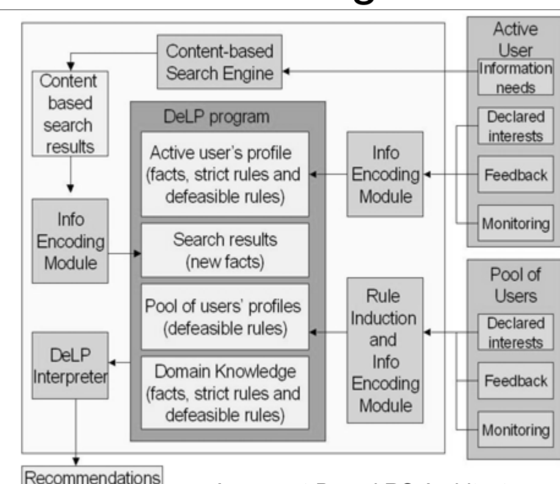
$$\{water\_garden(today)\} \xleftarrow{watergarden} \{\sim rain(today)\},\ not\,\{rain(X)\}$$

See *http://www.pims.math.ca/science/2004/NMR/ac.html* or *http://cs.uns.edu.ar/~grs*

Work in Progress

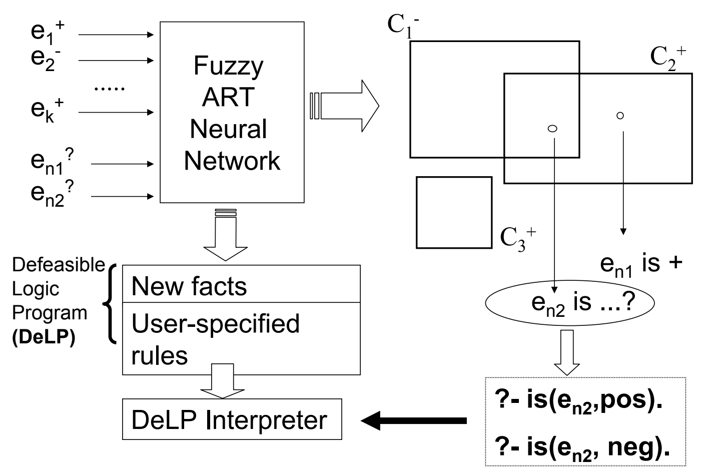
- Implementation issues considering world dynamics.
- The set of agent's beliefs is formed by the warranted literals, *i.e.*, those literals that are supported by an undefeated argument.
- As an agent receive new perceptions, beliefs could change.
- Because the process of calculating the new warrants is computationally hard we have developed a system to integrate precompiled knowledge in DeLP to address real time constrains for belief change. Our goal is to avoid re-computing arguments.
- See *http://web.dis.unimelb.edu.au/pgrad/iyadr/argmas/* or *http://cs.uns.edu.ar/~grs*

Work in Progress



Argument-Based RS Architecture

Work in Progress



FLAIRS 2004 *http://fermat.eps.udl.es/~cic/*

Belief Revision and Defeasible Reasoning

Belief Revision

What is the motivation of belief revision?

To model the Dynamics of Knowledge

How can we do that?

Classical Logic

+ Selection Mechanism

Non-classical Logic

An Example

From the following beliefs

The bird caught in the trap is a swan

The bird caught in the trap comes from Sweden

Sweden is part of Europe

All European swans are white

It can be inferred that

The bird caught in the trap is white

Now, new information arrives:

The bird caught in the trap is black

What it should be thrown away?

(Example due Peter Gärdenfors and Hans Rott, Belief Revision. Handbook of Logic in Artificial Intelligence and Logic Programming, vol. 4,1995)

Epistemic Models

- *Belief Sets*:
  Sets of sentences closed under logical consequence.
- *Belief Bases*:
  Arbitrary sets of sentences.

Epistemic Attitudes

Let K be a consistent belief base and let α be a sentence.

- α is *accepted* when $\alpha \in Cn(K)$
- α is *rejected* when $\sim\alpha \in Cn(K)$
- α is *indetermined* when $\alpha \notin Cn(K)$ and $\sim\alpha \notin Cn(K)$

If K is inconsistent then every sentence is accepted (and rejected).

Operations

Expansion ($+$): Allows to transform *indetermined* senteces in *accepted* or *rejected*:

a) If is α indetermined in K then α is accepted in $K{+}\alpha$
b) If is α indetermined in K then α is rejected in $K{+}{\sim}\alpha$

Contraction ($\dot{-}$): Allows to transform *accepted* or *rejected* sentences in *indeterminded*:

a) If is α accepted in K then α is indetermined in $K\dot{-}\alpha$
b) If is α rejected in K then α is indetermined in $K\dot{-}{\sim}\alpha$

Revision ($*$): Allows to transform sentencias *accepted* in *rejected* and to transform *rejected* sentences in *accepted*:

a) If is α accepted in K then α is rejected in $K{*}{\sim}\alpha$
b) If is α rejected in K then α is accepted in $K{*}\alpha$

Operations

Expansion ($+$):

- $K{+}\alpha = Cn(K \cup \{ \alpha \})$ (*Belief Sets*)
- $K{+}\alpha = K \cup \{ \alpha \}$ (*Belief Bases*)

$\left.\begin{array}{l}\text{Contraction } (\dot{-}) \\ \text{Revision } (*)\end{array}\right\}$ *How can they be defined*?

Two possibilities have been introduced:

- Levi Identity: $K{*}\alpha = (K\dot{-}{\sim}\alpha){+}\alpha$
- Harper Identity: $K\dot{-}\alpha = K \cap K{*}{\sim}\alpha$

Contraction Postulates

Let K be a Belief Set.

Closure: $K \dot{-} \alpha$ is a belief set.

Inclusion: $K \dot{-} \alpha \subseteq K$

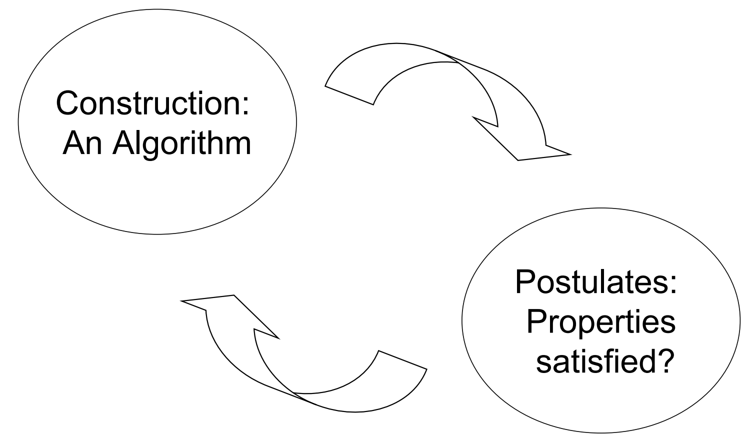
Vacuity: if $\alpha \notin K$ then, $K \dot{-} \alpha = K$

Success: if $\nvdash \alpha$ then $\alpha \notin K \dot{-} \alpha$

Recovery: if $\alpha \in K$ then, $K \subseteq (K \dot{-} \alpha) + \alpha$

Equivalence: if $\vdash \alpha \leftrightarrow \beta$, then $K \dot{-} \alpha = K \dot{-} \beta$

Change Operators



Partial Meet Contraction

Construction:

- ➡ $K \perp \alpha = \{H\colon H \subseteq K,\ \alpha \notin Cn(H)$ and for all $H \subset H' \subseteq K$ then $\alpha \in Cn(H')\}$
- ➡ $K \dot{-} \alpha = \bigcap \gamma(K \perp \alpha)$ → Selection Function
  - ☛ $\gamma(K \perp \alpha) \subseteq K \perp \alpha$
  - ☛ if $K \perp \alpha \neq \varnothing$, then $\gamma(K \perp \alpha) \neq \varnothing$ otherwise $\gamma(K \perp \alpha) = K$

Example:

- ➡ $K = \{\ a,\ b,\ a \wedge b \rightarrow c,\ d\ \}$
- ➡ $K \perp c = \{K_1, K_2, K_3\} = \{\{\ a,\ b,\ d\ \}, \{\ a,\ a \wedge b \rightarrow c, d\ \}, \{\ b,\ a \wedge b \rightarrow c, d\ \}\}$
- ➡ Some possible results of $K \dot{-} c$:

$\{\ a,\ b,\ d\ \}$	$\gamma(K \perp c) = \{\ K_1\ \}$
$\{\ a,\ d\ \}$	$\gamma(K \perp c) = \{\ K_1, K_2\ \}$
$\{\ a \wedge b \rightarrow c,\ d\ \}$	$\gamma(K \perp c) = \{\ K_2, K_3\ \}$
$\{\ d\ \}$	$\gamma(K \perp c) = \{\ K_1, K_2, K_3\ \}$

Kernel Contraction

Kernel mode:

- ➡ Let K be a set of sentences and α be a sentence.
- ➡ We found all minimal subsets of K implying α (called α-*kernels*).
- ➡ We "cut" the α-*kernels* by means of an incision function σ and then we eliminate the cut set from K.

Kernel Contraction

Construction:

- ➡ $K \perp\!\!\!\perp \alpha = \{H\colon H \subseteq K,\ \alpha \in Cn(H)$ and for all $H' \subset H$ then $\alpha \notin Cn(H')\}$
- ➡ $K \dot{-} \alpha = K \setminus \sigma(K \perp\!\!\!\perp \alpha)$

Incision Function:

- $\sigma(K \perp\!\!\!\perp \alpha) \subseteq \bigcup K \perp\!\!\!\perp \alpha$
- if $x \in K \perp\!\!\!\perp \alpha$ and $x \neq \varnothing$, then $x \cap \sigma(K \perp\!\!\!\perp \alpha) \neq \varnothing$

Example:

- ➡ $K = \{\ a,\ a \rightarrow c,\ b,\ b \rightarrow c,\ d,\ \sim e\ \}$
- ➡ $K \perp\!\!\!\perp c = \{\{\ a,\ a \rightarrow c\ \},\ \{\ b,\ b \rightarrow c\ \}\}$
- ➡ Some possible results of $K - c$:

$\{\ a \rightarrow c,\ b \rightarrow c,\ d,\ \sim e\ \}$	$\sigma(K \perp\!\!\!\perp c) = \{\ a,\ b\ \}$
$\{\ a,\ b \rightarrow c,\ d,\ \sim e\ \}$	$\sigma(K \perp\!\!\!\perp c) = \{\ a \rightarrow c,\ b\ \}$
$\{\ b \rightarrow c,\ d,\ \sim e\ \}$	$\sigma(K \perp\!\!\!\perp c) = \{\ a,\ a \rightarrow c,\ b\ \}$
$\{\ d,\ \sim e\ \}$	$\sigma(K \perp\!\!\!\perp c) = \{\ a,\ a \rightarrow c,\ b,\ b \rightarrow c\ \}$

Controversial Postulates

- ➡ Every construction of a change operator is charaterized by postulates.
- ➡ In the *AGM* model, there are some controversial postulates.
- ➡ Contraction:
  - *Recovery*: $K \subseteq (K \dot{-} \alpha) + \alpha$
- ➡ Revision:
  - *Success*: $\alpha \in K * \alpha$
  - *Consistency*: If α is consistent then $K * \alpha$ is consistent.

Explanations, Belief Revision and Defeasible Reasoning

Belief Bases

There are two kinds of beliefs:

- ➡ *Explicit Beliefs*: all the sentences *in* the belief base.
- ➡ *Implicit Beliefs*: all sentences *derived* from the belief base.

> The implicit beliefs are *“explained”* from more basic beliefs.

Explanations

An *explanans* justifies an *explanandum*.

Set of sentences A sentence

Notation: $A \rightarrowtail \alpha$

Properties:

- *Deduction*: $A \vdash \alpha$
- *Consistency*: It is not the case that $A \nvdash \bot$
- *Minimality*: There is no set $A' \subset A$ such that $A' \vdash \alpha$
- *Informational Content*: It is not the case that $\alpha \vdash A$

Informational Content

- It is not the case that $\alpha \vdash A$

This postulate precludes the following cases:

Self-explanation:

$$\{ \alpha \} \rightarrowtail \alpha$$

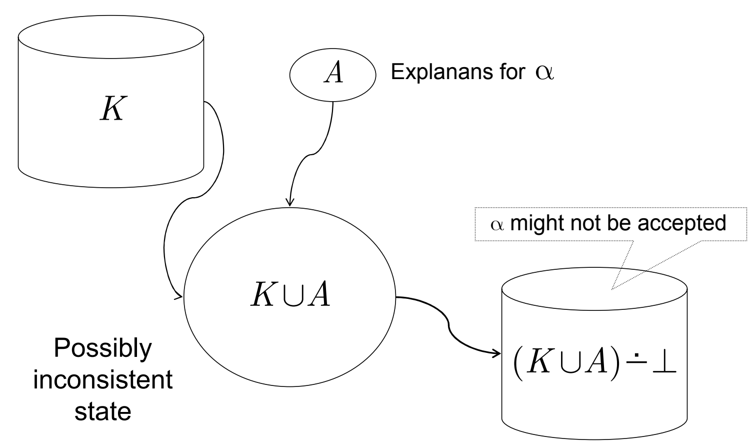
Redundancy:

$$\{ \alpha \vee \beta, \alpha \vee \sim\beta \} \rightarrowtail \alpha$$

New change operators

- We will define operators for revision with respect to an explanans (*i.e.,* a set of sentences).
- The idea is the following:
  - Instead of incorporating a sentence α we request an explanans A for α.
  - We add A to K
  - Then, we restore consistency (Consolidation).

New change operators



Two kinds of Constructions

- Partial Meet Revision by a set of sentences:

$$K \circ A = (K \cup A) \underbrace{\dot{-}}_{\text{Partial Meet Contraction Operator}} \perp$$

- Kernel Revision by a set of sentences:

$$K \circ A = (K \cup A) \underbrace{\dot{-}}_{\text{Kernel Contraction Operator}} \perp$$

G.R.Simari, ICLP 2004 101

Different kinds of beliefs

- Particular Beliefs:

$car(ferrari)$ $bird(opus)$

- General Beliefs:

$\forall x(car(x) \rightarrow vehicle(x))$ $\forall x(bird(x) \rightarrow flies(x))$

The strategy: all beliefs removed in a change process are preserved in a different status.

G.R.Simari, ICLP 2004 102

Transformation of Beliefs

Transf $(\ (\forall x)(p(x) \rightarrow q(x)) \ )$

$p(x) \succ\!\!- \ q(x)$

Defeasible rule in Argumentative Systems

or

$$\frac{p(x) \ : \ q(x)}{q(x)}$$

Default rule in Default Theories

G.R.Simari, ICLP 2004 103

Epistemic Model

A knowledge structure $[K, \Delta]$ where:

- K is the *undefeasible knowledge*.
- Δ is the *defeasible knowledge* represented by:
  - Defeasible conditionals in Argumentative Systems; or
  - Default rules in Default Theories.

G.R.Simari, ICLP 2004 104

Changes

$[K, \Delta]\circ A = [K', \Delta']$

where:

- $K' = K \circ A$
- $\Delta' = \Delta \cup \{ \mathit{Transf}(\alpha) : \alpha \in K \setminus K \circ A \}$

Example

- $K = \{ \mathit{bird}(\mathit{tweety}), \mathit{bird}(\mathit{opus}),$
  $\forall x(\mathit{peng}(x) \rightarrow \mathit{bird}(x)), \forall x(\mathit{bird}(x) \rightarrow \mathit{fly}(x))\}$
- From K we may conclude that:
  $\mathit{bird}(\mathit{tweety}), \mathit{bird}(\mathit{opus}), \mathit{fly}(\mathit{tweety}), \mathit{fly}(\mathit{opus})$
- Then, we receive the next explanans A for $\sim\mathit{fly}(\mathit{opus})$:
  $\{ \mathit{bird}(\mathit{opus}), \mathit{peng}(\mathit{opus}),$
  $\forall x(\mathit{peng}(x) \wedge \mathit{bird}(x) \rightarrow \sim\mathit{fly}(x) \}$

Example

- In order to obtain $K \circ A$ we need to eliminate contradictions from $K \cup A$.

  $K \cup A = \{ \mathit{bird}(\mathit{tweety}), \mathit{bird}(\mathit{opus}), \mathit{peng}(\mathit{opus}),$
  $\forall x(\mathit{peng}(x) \rightarrow \mathit{bird}(x)), \forall x(\mathit{bird}(x) \rightarrow \mathit{fly}(x)),$
  $\forall x(\mathit{peng}(x) \wedge \mathit{bird}(x) \rightarrow \sim\mathit{fly}(x)\}\}$
- We could give up particular or general beliefs.
- If we discard general beliefs, we could select the *less specific* beliefs, for instance, $\forall x(\mathit{bird}(x) \rightarrow \mathit{fly}(x))$.

Example

- Then, we have the following belief base:
  $K \circ A = \{ \mathit{bird}(\mathit{tweety}), \mathit{bird}(\mathit{opus}), \forall x(\mathit{peng}(x) \rightarrow \mathit{bird}(x)),$
  $\mathit{peng}(\mathit{opus}), \forall x(\mathit{peng}(x) \wedge \mathit{bird}(x) \rightarrow \sim\mathit{fly}(x)) \}$
- From $K \circ A$ me may conclude that:
  $\mathit{bird}(\mathit{tweety}), \mathit{bird}(\mathit{opus}), \mathit{peng}(\mathit{opus}), \sim\mathit{fly}(\mathit{opus})$
- We can't conclude $\mathit{fly}(\mathit{tweety})$ even though it is consistent with K.
- This problem can be solved if we preserve the defeasible conditional $\mathit{bird}(x) \succ\!\!- \mathit{fly}(x)$ or the default rule $\mathit{bird}(x) : \mathit{fly}(x) \,/\, \mathit{fly}(x)$.

Example

- That is, we have the following knowledge:

$$K \circ A = \{\ bird(tweety),\ bird(opus),\ peng(opus),\ \forall x(peng(x) \wedge bird(x) \rightarrow \sim fly(x))\}$$

$$\Delta = \{\ bird(x) \succ fly(x)\ \}$$

- From $[K \circ A\ ,\ \Delta]$ we can infer that:

$$bird(tweety),\ bird(opus),\ peng(opus),\ \sim fly(opus),\ fly(tweety)$$

- We have a new epistemic model and a new set of epistemic attitudes.

G.R.Simari, ICLP 2004 109

Two Interesting Surveys

- *Logical Systems for Defeasible Argumentation*, H. Prakken, G. Vreeswijk, in D. Gabbay (Ed.), Handbook of Philosophical Logic, 2nd Edition, 2000.
- *Logical Models of Argument*, C. I. Chesñevar, A. G. Maguitman, R. P. Loui, ACM Computing Surveys, **32**(4), pp 337-383, 2000.

G.R.Simari, ICLP 2004 110

References for the work presented (Short List)

- *Defeat Among Arguments: A System of Defeasible Inference,* R. P. Loui, Computational Intelligence, Vol 3, 3, 1987.
- *Defeasible Reasoning*, J. Pollock, Cognitive Science, **11**, 481-518, 1987.
- *Defeasible Reasoning: A Philosophical Analysis in PROLOG,* Donald Nute, in J. H. Fetzer (Ed.) Aspects of Artificial Intelligence, 1988.
- *A Mathematical Treatment of Defeasible Reasoning and Its Implementation*, G. R. Simari, R. P. Loui, Artificial Intelligence, **53**, 125-157, 1992.
- *An Argumentation Semantics for Logic Programming with Explicit Negation.* P. M. Dung, in Proceedings 10th. Intenational Conference on Logic Programming, 616-630, 1993.

G.R.Simari, ICLP 2004 111

References for the work presented (Short List)

- *Cognitive Carpentry: A Blueprint for How to Build a Person*. J. Pollock. MIT Press, 1995.
- *An Abstract, Argumentation-Theoretic Approach to Default Reasoning*, A. G. Bondarenko, P. M. Dung, R. A. Kowalski, F. Toni, Artificial Intelligence (93), 1-2, 63-101, 1997.
- *Abstract Argumentation Systems*, G. Vreeswijk, Artificial Intelligence, **90,** 225-279, 1997.
- *Explanations, Belief Revision and Defeasible Reasoning,* M. Falappa, G. Kern-Isberner, G. R. Simari, Artificial Intelligence **141** (2002) 1-28.
- *Defeasible Logic Programming: An Argumentative Approach,* A. J. García, G.R. Simari, Theory and Practice of Logic Programming. Vol 4(1), 95-138, 2004.

G.R.Simari, ICLP 2004 112