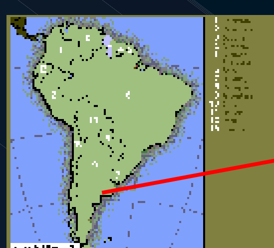
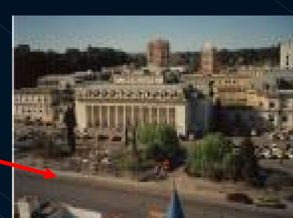
Computational Models for Argumentation in MAS

Carlos Iván Chesñevar
Dept. of Computer Science
UNIVERSITAT DE LLEIDA
SPAIN

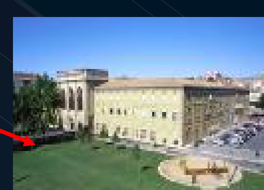
Guillermo R. Simari
Dept. of Computer Science and Engineering
UNIVERSIDAD NACIONAL DEL SUR
ARGENTINA

Where are we from…





Univ. Nacional del Sur
(Bahía Blanca, Argentina)



University of Lleida
(Lleida, Catalonia, Spain)

Main references

- H. Prakken, G. Vreeswijk. *Logical Systems for Defeasible Argumentation*, in D. Gabbay (Ed.), Handbook of Philosophical Logic, 2nd Edition, 2002.
- C.Chesñevar, A.Maguitman, R.Loui. *Logical Models of Argument*. In ACM Computing Surveys, Dec. 2000.
- I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg (2003b) *"Argumentation-based negotiation"*. The Knowledge Engineering Review 18 (4) 343-375.

Outline

- (Very brief) Introduction to Multiagent Systems
- What is argumentation? Fundamentals
- A Case Study: DeLP and its extensions as an argument-based approach to logic programming.
- Argumentation meets agents: argument-based negotiation
- Conclusions

Overview

- Five ongoing trends have marked the history of computing:
  - *ubiquity*;
  - *interconnection*;
  - *intelligence*;
  - *delegation*; and
  - *human-orientation*

Credits: some of these slides are based on Michael Wooldridge's lecture notes for his book "An Introduction to MAS" (Wiley & Sons, 2002)

5

Ubiquity, Interconnection, Intelligence

- As processing capability spreads, sophistication (and intelligence of a sort) becomes ubiquitous.
- What could benefit from having a processor embedded in it…?
- Internet is powerful…Some researchers are putting forward theoretical models that portray computing as primarily a process of interaction.
- The complexity of tasks that we are capable of automating and delegating to computers has grown steadily.

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 6

Delegation, Human-Orientation

- Computers are doing more for us – without our intervention. Next on the agenda: fly-by-wire cars, intelligent braking systems…
- Programmers conceptualize and implement software in terms of higher-level – more human-oriented – abstractions.
- The movement away from machine-oriented views of programming toward concepts and metaphors that more closely reflect the way we ourselves understand the world.

Programming progression…

- Programming has progressed through:
  - machine code;
  - assembly language;
  - machine-independent programming languages;
  - sub-routines;
  - procedures & functions;
  - abstract data types;
  - objects;

  to *agents*.

Where does it bring us?

- Delegation and Intelligence imply the need to build computer systems that can act effectively on our behalf.
- This implies:
  - The ability of computer systems to act *independently*.
  - The ability of computer systems to act in a way that *represents our best interests* while interacting with other humans or systems.

Interconnection and Distribution

- Interconnection and Distribution have become core motifs in Computer Science.
- But Interconnection and Distribution, coupled with the need for systems to represent our best interests, implies systems that can *cooperate* and *reach agreements* (or even *compete*) with other systems that have different interests (much as we do with other people).

So Computer Science expands…

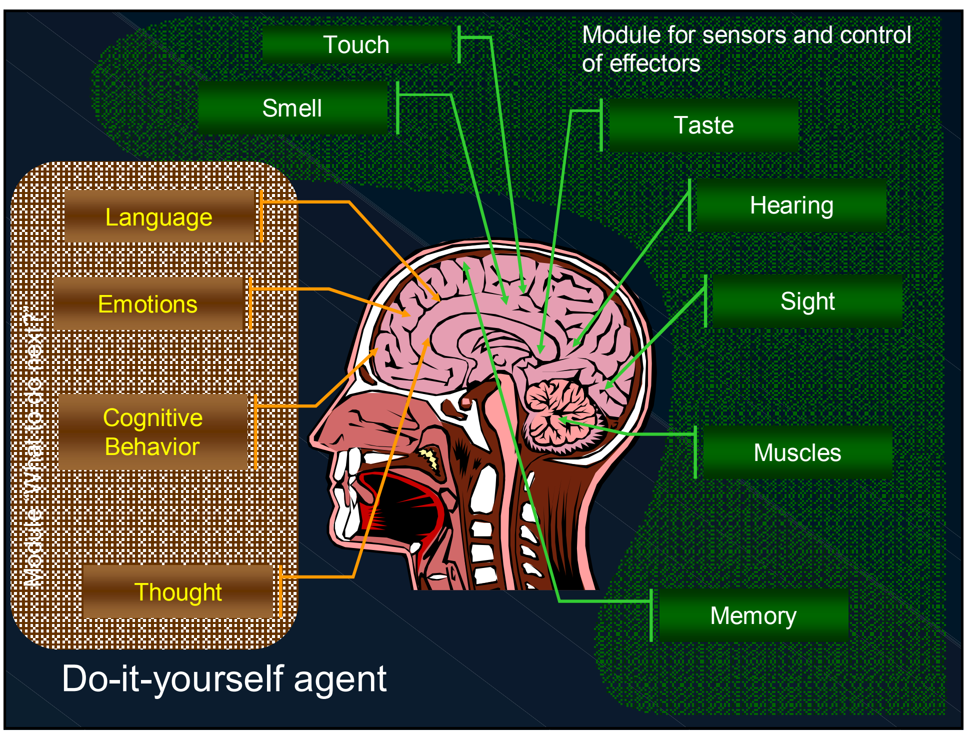
- These issues were not studied in Computer Science until recently.
- All of these trends have led to the emergence of a new field in Computer Science: ***Multiagent Systems.***
- An agent is a computer system that is capable of *independent* action on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told).

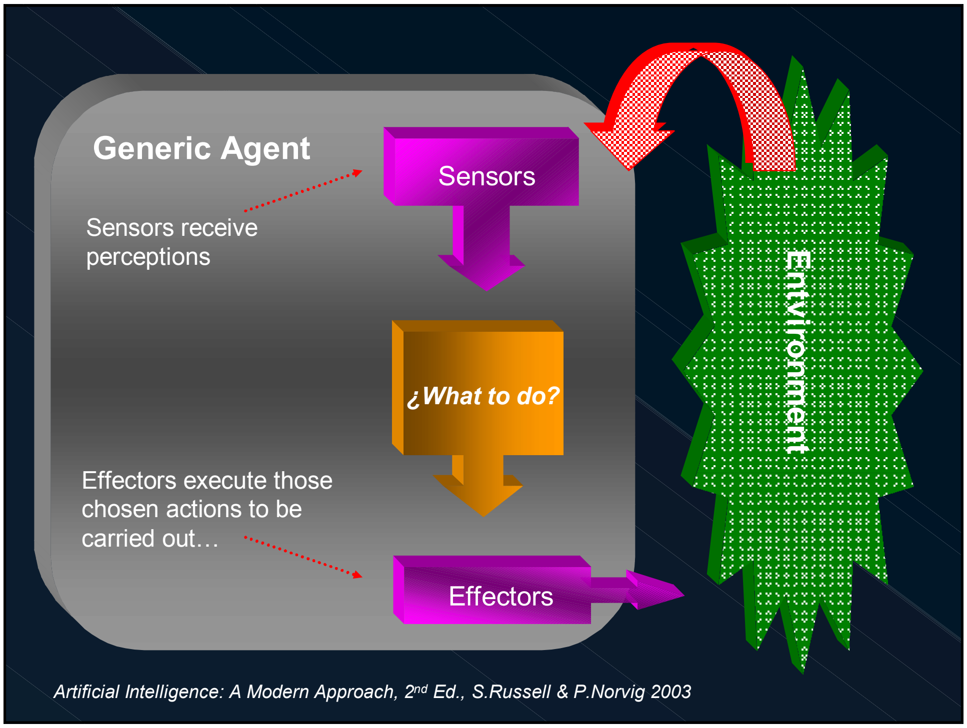
Multiagent Systems: a Definition

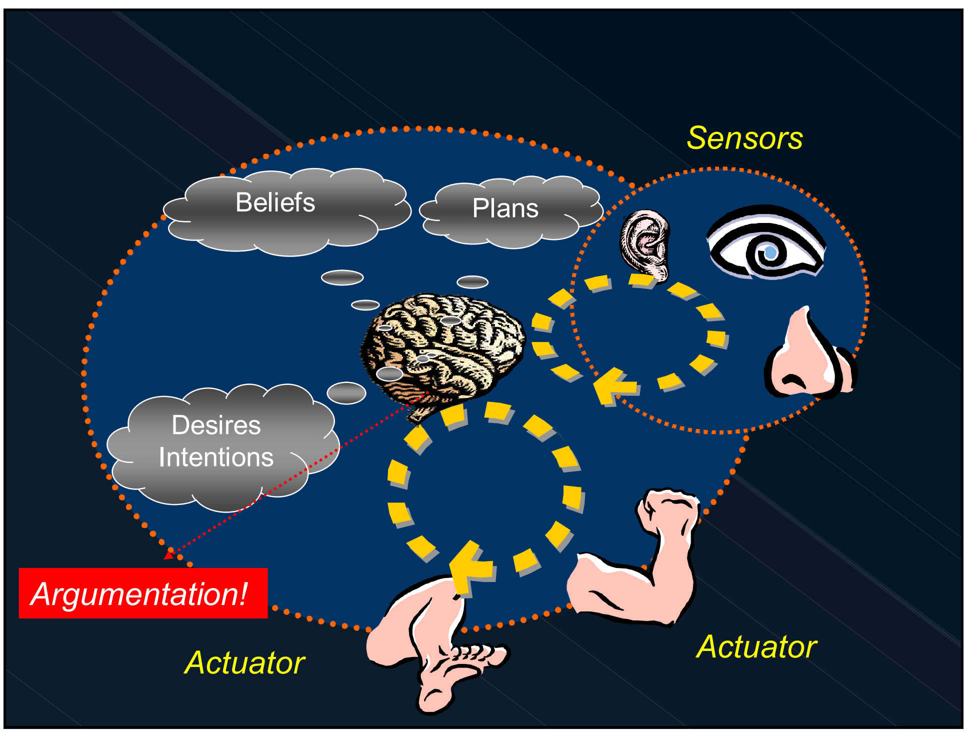
- A multiagent system is one that consists of a number of agents, which *interact* with one-another.
- In the most general case, agents will be acting on behalf of users with different goals and motivations.
- To successfully interact, they will require the ability to *cooperate*, *coordinate*, and *negotiate* with each other, much as people do.

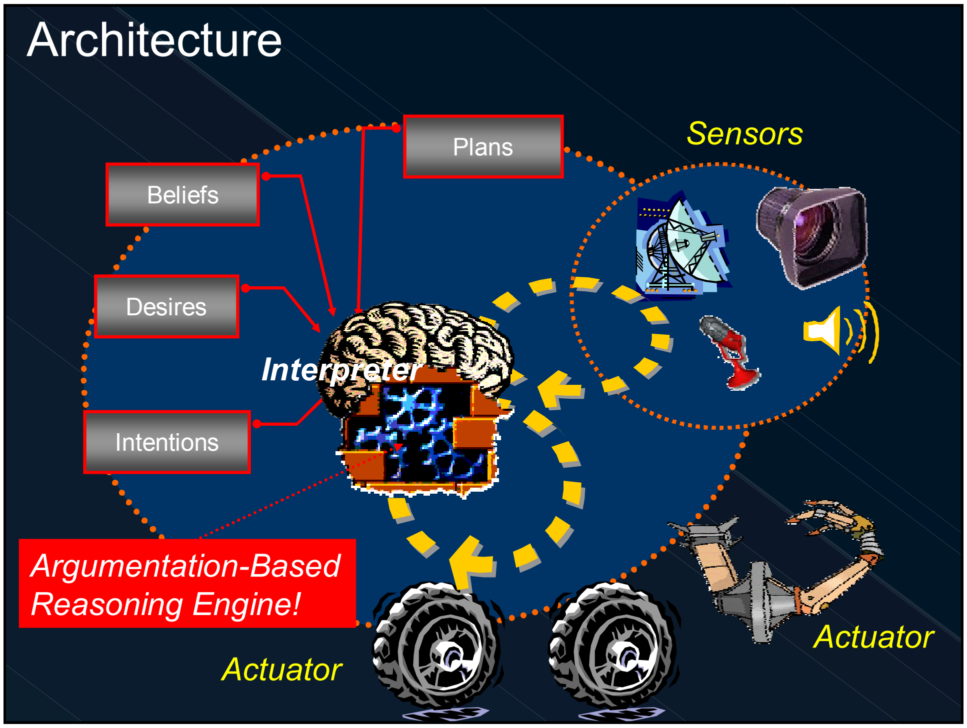
Multiagent Systems

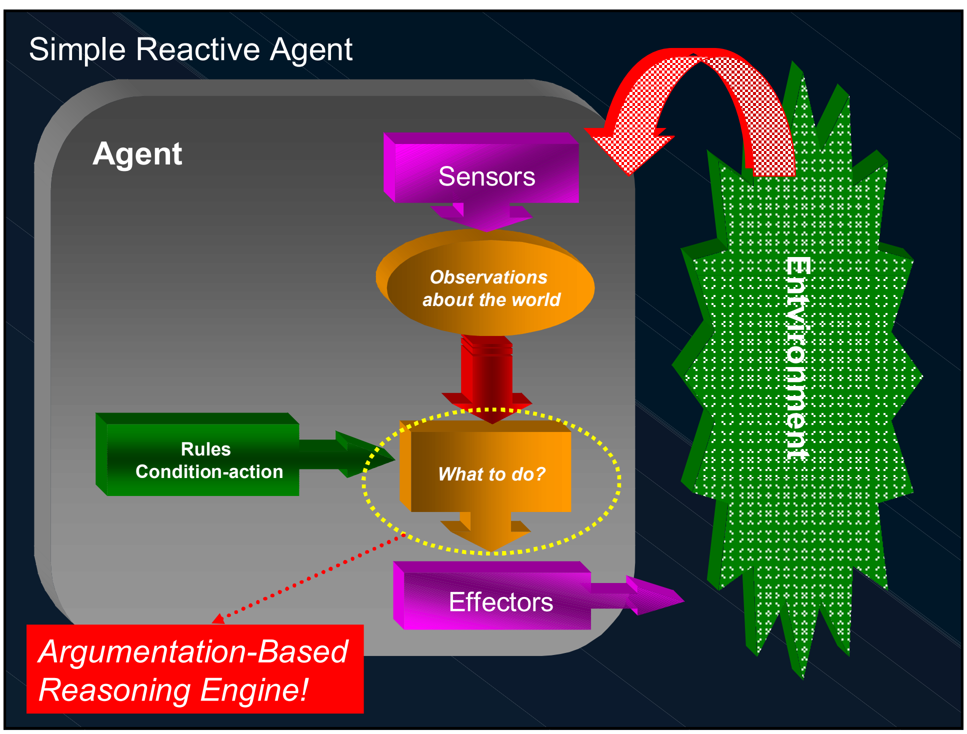
- In Multiagent Systems, we address questions such as:
  - How can cooperation emerge in societies of self-interested agents?
  - What kinds of languages can agents use to communicate?
  - How can self-interested agents recognize conflict, and how can they (nevertheless) reach agreement?
  - How can autonomous agents coordinate their activities so as to cooperatively achieve goals?

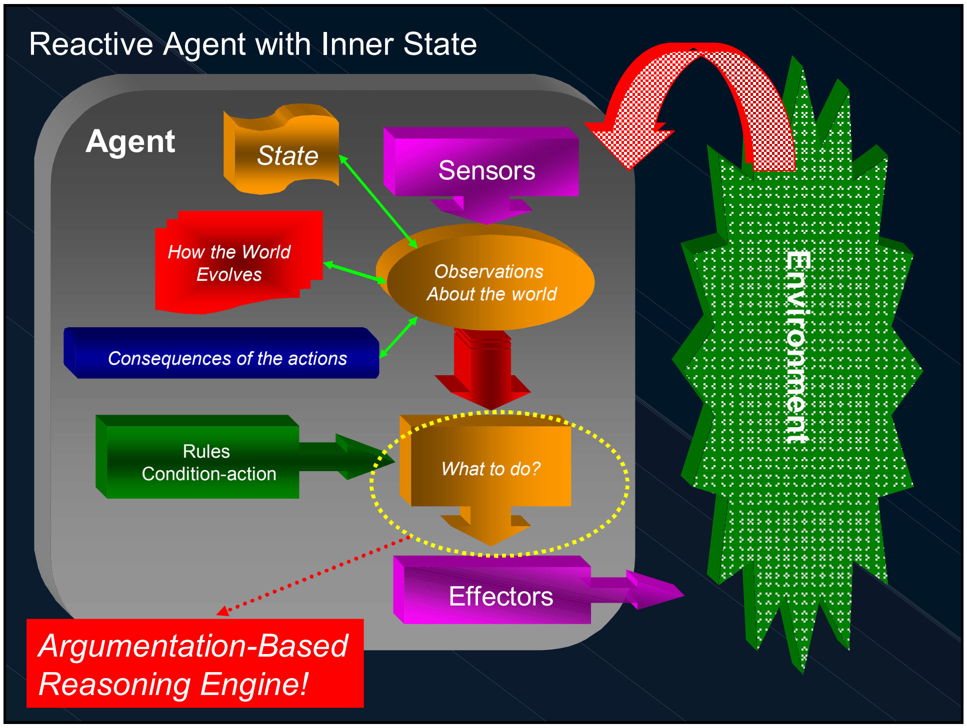


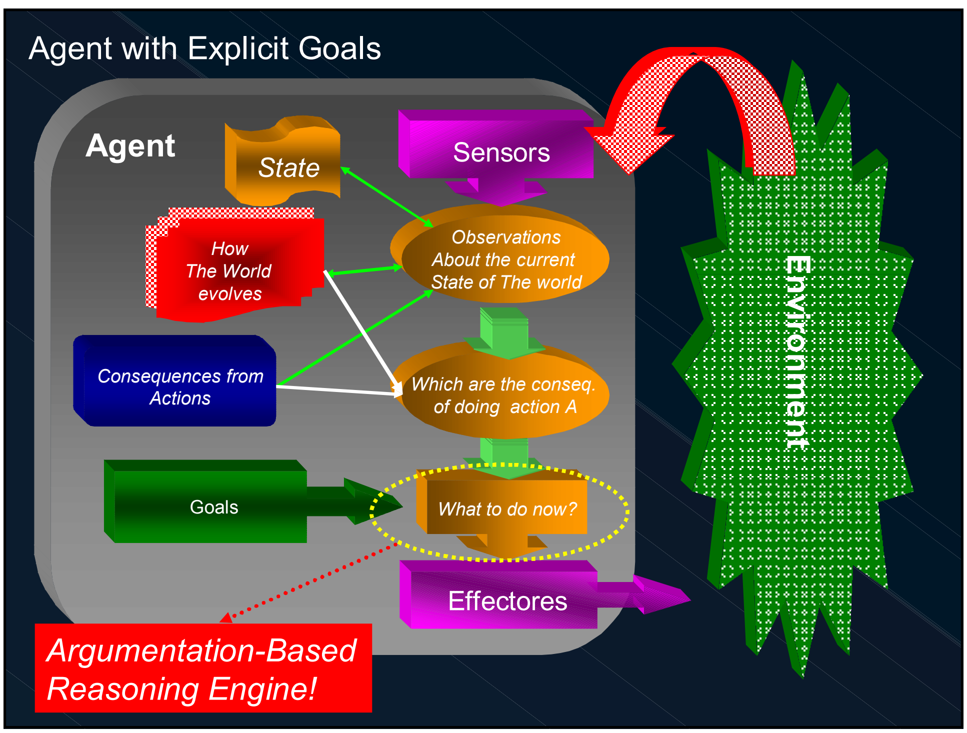

Generic Agent
Sensors
Sensors receive perceptions
¿What to do?
Effectors execute those chosen actions to be carried out…
Effectors
Environment
Artificial Intelligence: A Modern Approach, 2nd Ed., S.Russell & P.Norvig 2003

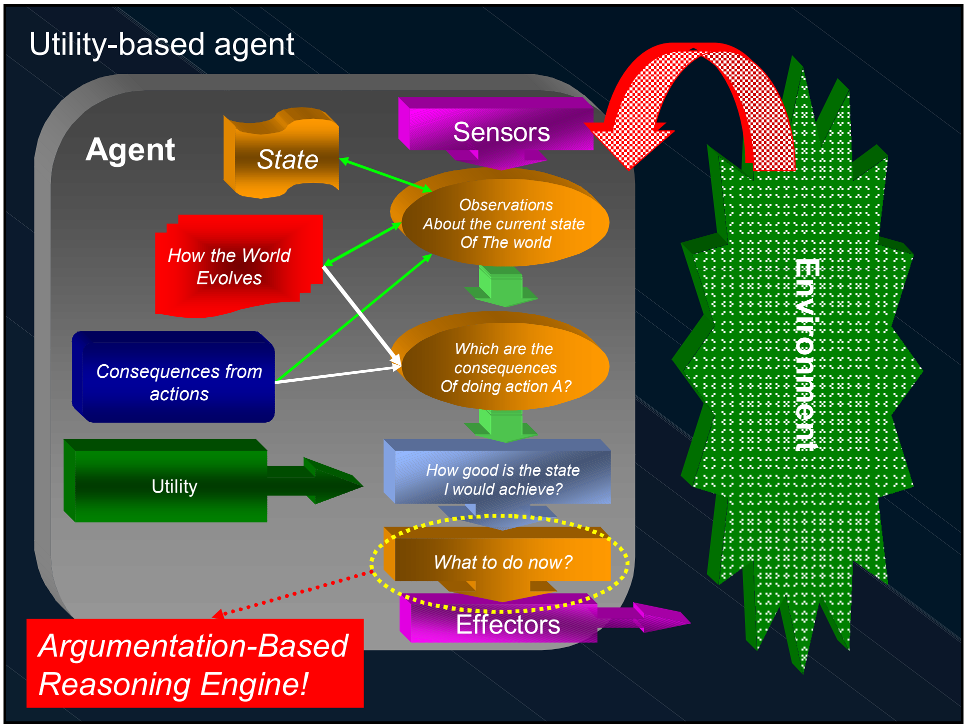

Sensors
Beliefs
Plans
Desires
Intentions
Argumentation!
Actuator
Actuator


Architecture
Plans
Sensors
Beliefs
Desires
Interpreter
Intentions
Argumentation-Based
Reasoning Engine!
Actuator
Actuator


Simple Reactive Agent
Agent
Sensors
Observations
about the world
Rules
Condition-action
What to do?
Effectors
Environment
Argumentation-Based
Reasoning Engine!


Reactive Agent with Inner State
Agent
State
Sensors
How the World Evolves
Observations About the world
Consequences of the actions
Rules Condition-action
What to do?
Effectors
Environment
Argumentation-Based Reasoning Engine!


Agent with Explicit Goals
Agent
State
Sensors
How The World evolves
Observations About the current State of The world
Consequences from Actions
Which are the conseq. of doing action A
Goals
What to do now?
Effectores
Environment
Argumentation-Based Reasoning Engine!



Outline

- (Very brief) Introduction to Multiagent Systems
- What is argumentation? Fundamentals
- A Case Study: DeLP and its extensions as an argument-based approach to logic programming.
- Argumentation meets agents: argument-based negotiation
- Conclusions

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 22

Systems for defeasible argumentation. Generalities

Typical problems in (non-monotonic) default reasoning:

1) Representation of defaults: *e.g.* Birds usually fly

2) Inconsistency handling: identify relevant subsets of consistent information.

3) Identifying preferred models

Many approaches have been developed:

- Default logic (Reiter, 1980)
- Preferred subtheories (Brewka, 1989)
- Circunscription (McCarthy, 1987)
- Others…

Systems for defeasible argumentation. Generalities

Argumentation systems (AS) are "yet another way" to formalize common-sense reasoning. Non-monotonicity arises from the fact that new premises may give rise to stronger counterarguments, which in turn will defeat the original argument.

1) Normality condition view: an argument = standard proof from a set of premises + normality statements. A counterargument is an attack on such a normality statement.

2) Inconsistency handling view: an argument = standard proof from a consistent subset of the premises. A counterargument is an attack on a premise of an argument.

3) Semantic view: constructing 'invalid' arguments (wrt the semantics) is allowed in the proof theory. A counterargument is an attack on the use of an inference rule which deviates from a preferred model.

Views on default reasoning from an argumentation perspective

Systems for Defeasible Argumentation

According to Prakken & Vreeswijk (2002), there are five common elements to systems for defeasible argumentation:

Definition of Underlying Logical Language
Definition of Argument
Definition of Conflict among Arguments
Definition of Defeat among Arguments
Definition of Status of Arguments

25

The underlying logic: Arguments & Logical consequence

- Argumentation Systems are constructed starting from a *logical language* and an associated notion of *logical consequence* for that language.
- The logical consequence relation helps to define what will be considered an *argument*.
- This consequence relation is *monotonic*, *i.e.,* new information cannot invalidate arguments as such, but rather give rise to counterarguments.
- Arguments are seen as proofs in the chosen logic.

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 26

Argument as a 'proof'

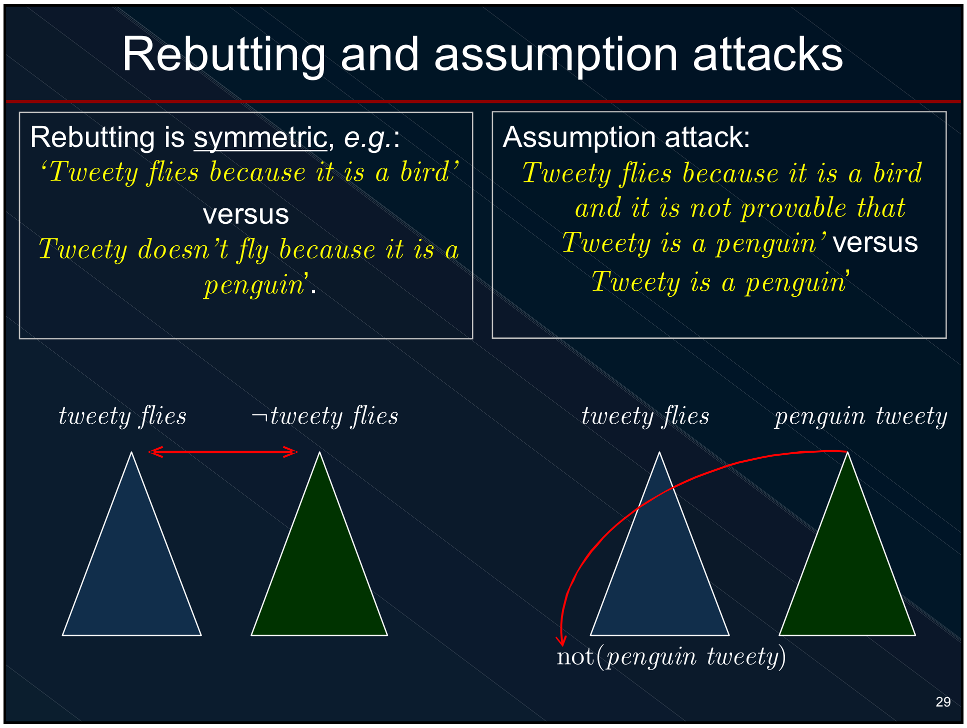
Arguments are presented under different forms:

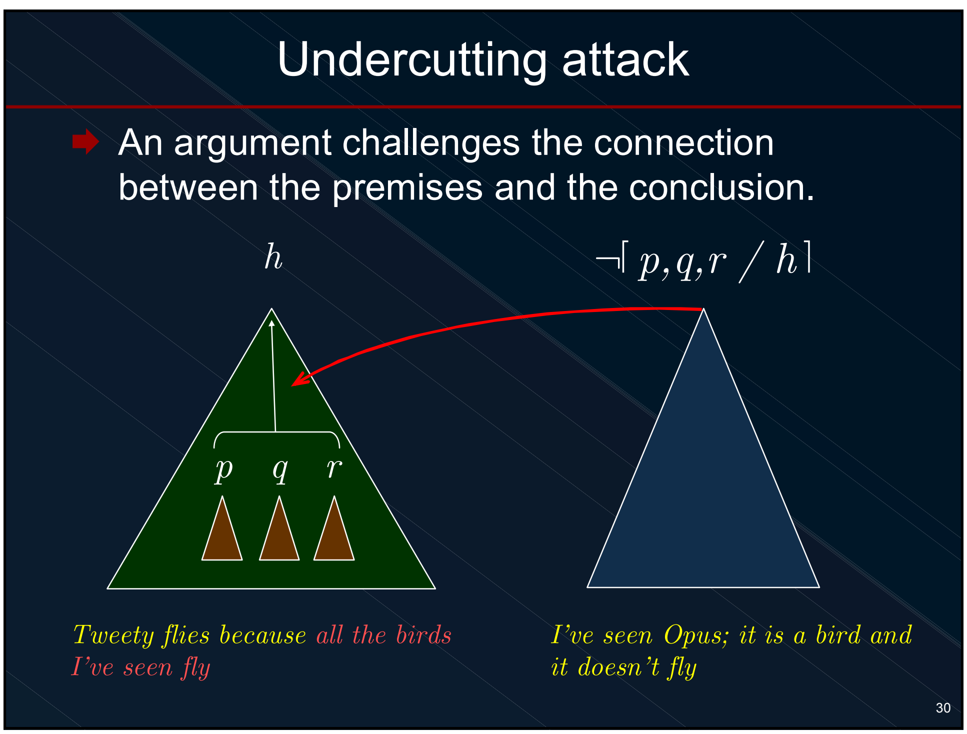
- An inference tree grounded in premises.
- A deduction sequence.
- A pair (*Premises, Conclusion*), leaving unspecified the particular proof, in the underlying logic, that leads from the *Premises* to the *Conclusion*.
- A completely unspecified structure, such as in Dung's abstract framework for argumentation (1995).

Conflict, Attack, Counterargument

The notion of conflict (Counterargument or Attack) between arguments is typically discussed discriminating three cases:

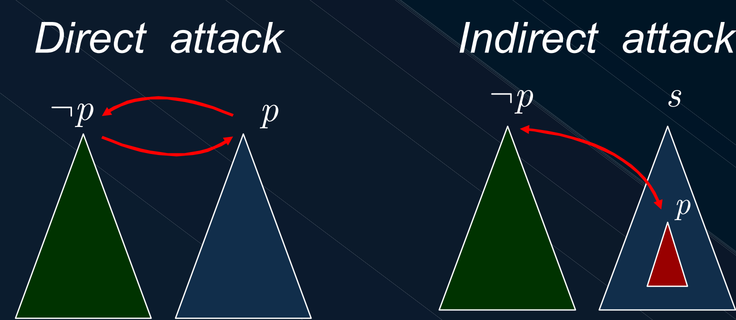
- *Rebutting attacks*: arguments with contradictory conclusions.
- *Assumption attack*: attacking non-provability assumptions.
- *Undercutting attacks*: an argument that undermines some intermediate step (inference rule) of another argument.


Rebutting and assumption attacks
Rebutting is symmetric, e.g.:
'Tweety flies because it is a bird'
versus
Tweety doesn't fly because it is a penguin'.
Assumption attack:
Tweety flies because it is a bird and it is not provable that Tweety is a penguin' versus
Tweety is a penguin'
tweety flies
¬tweety flies
tweety flies
penguin tweety
not(penguin tweety)
29


Undercutting attack
An argument challenges the connection between the premises and the conclusion.
h
¬⌈ p,q,r / h⌉
p
q
r
Tweety flies because all the birds I've seen fly
I've seen Opus; it is a bird and it doesn't fly
30

Direct vs. Indirect Attack

These types of attack could be *direct* and *indirect*.

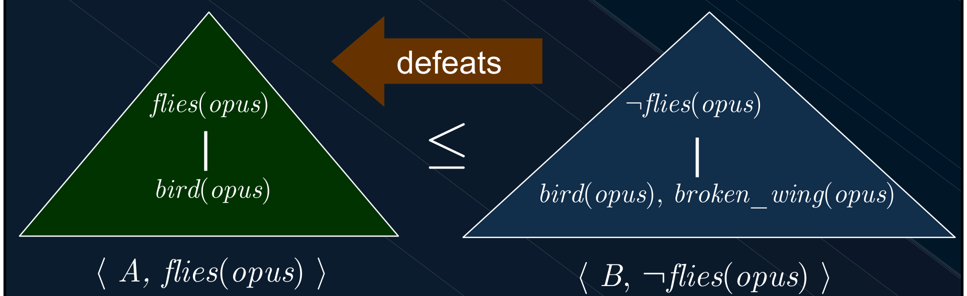


Defeat: Comparing Arguments

- The notion of conflict does not embody any form of *comparison*; this is another element of AS.
- *Defeat has the form of a binary relation between arguments*, standing for
  - ‘*attacking and not weaker*’ (defeat)
  - ‘*attacking and stronger*’ (strict defeat)
- Terminology varies: ‘*defeat*’ (Simari, 1989; Prakken & Sartor, 1997), ‘*attack*’ (Dung, 1995; Bondarenko *et. al* 1997) and ‘*interference*’ (Loui, 1998).

Defeat: Comparing Arguments

- Argumentation systems vary in their grounds for evaluation of arguments. One common criterion is the *specificity principle*, which prefers arguments based on the most specific defaults.



33

Defeat: Comparing Arguments

- However, it has been argued that specificity is not a general principle of commonsense reasoning, but rather a standard that might (or might not) be used.
- Some researchers even claim that general, domain-independent principles of defeat *do not exist*, or are very weak.
- Some even argue that the evaluation criteria are part of the domain theory, and should also be debatable.

What do you think?

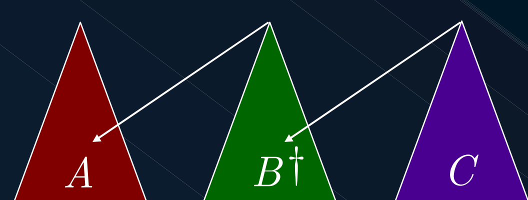
Computational Models for Argumentation in Multiagent Systems – EASSS 2005 34

Defeat: comparing arguments

- In Simari&Loui's framework, specificity is used as a default, but it is 'modular': any other preference relation defined among arguments could be used.
- In Dung's, defeat is an abstract notion, left undefined.
- In Bondarenko's framework, defeat is limited to attack between arguments (there is no preference at all!)
- Other comparison criteria are possible…

Defeat: comparing arguments

- Defeat is basically a binary relation on a set of args.
- But ... it just tells us something about <u>two</u> arguments, not about a dispute (that may involve many args.)
- A common situation is *reinstatement* as in the example below (where an argument C reinstates an argument A by defeating argument B)

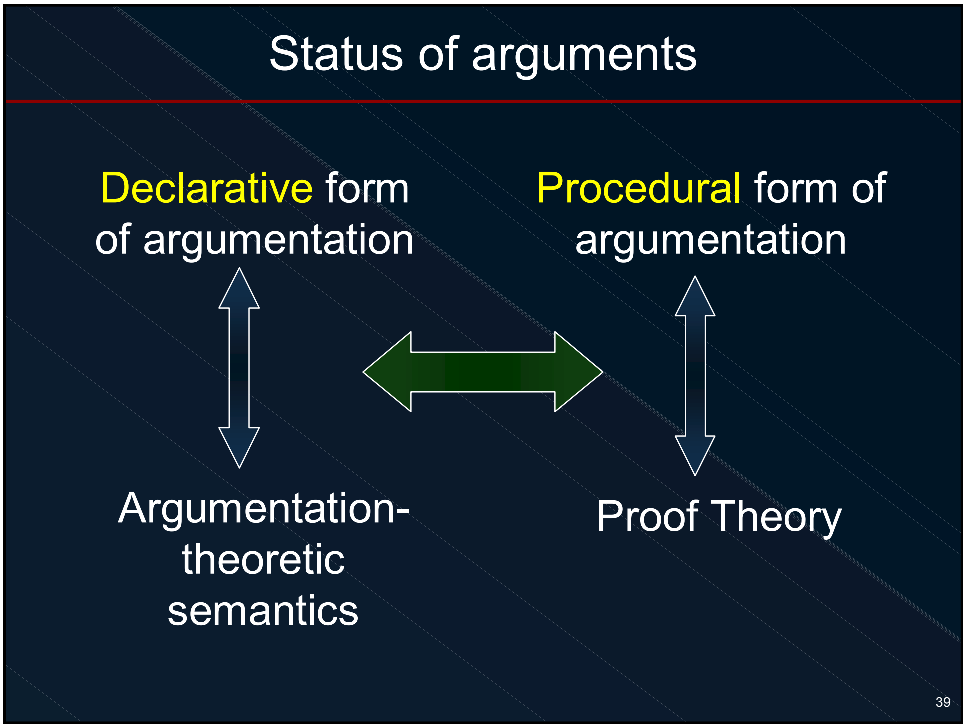


Status of Arguments

- The last element in our ontology comes into play... the definition of *Status of Arguments*.
- This notion is the actual output of most Arg.Sys and arguments are divided into (at least) two classes:
  - Arguments with which a dispute can be *'won'*
  - Arguments with which a dispute can be *'lost'*
  - Arguments that leave the dispute *'undecided'*
- Usual terminology: *'justified'* or *'warranted'* *vs.* *'defeated'* or *'overruled'* *vs.* *'defensible'*, *etc*.

Status of arguments

- Status of arguments can be computed either in '*declarative'* or '*procedural*' form.
- In the declarative form usually requires fixed-point definitions, and establishes certain sets of arguments as acceptable (in the context of a set of premises and a evaluation criteria) but without defining a procedure for testing whether a given argument is a member of this set.
- *'Procedural form'* amounts to defining such a procedure for acceptability.



Model-theoretic Semantics

- Default logic was initially criticized by the lack of a model-theoretic semantics...
- Several researchers argued that NMR needs a different kind of semantics than model theory suggesting an argumentation-theoretic semantics.
- Model theory provides meaning to logical languages by defining how the world would be if an expression with these symbols would be true.
- *Should this be the case for argumentative systems ...?*

Model-theoretic Semantics

- Some researchers (e.g. Pollock, Vreeswijk, Loui) argue that the meaning of defaults should not be found in a correspondence with reality, but in their role in *dialectical inquiry*.
- This approach goes as follows: *since the central notions of defeasible reasoning are not propositional, then the semantics should also be different, i.e., an argumentation-theoretic semantics should be defined.*

Argumentation-theoretic Semantics

- Defeasible rules "$premises \Rightarrow conclusion$" induce a *burden of proof*, rather than a correspondence between a proposition and the world.
- Argumentation-theoretic semantics tries to capture sets of arguments that are as large as possible, and defend themselves against attacks on their members.

Argument-based Semantics

- Which conditions on sets of arguments should be satisfied?
- We will assume as background
  - A set $Args$ of arguments
  - A binary relation of *'defeat'* defined over it.

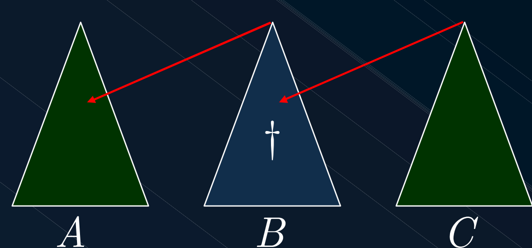
Def. 1: Arguments are either *justified* or *not justified*

1. An argument is justified if all arguments defeating it (if any) are not justified.
2. An argument is not justified if it is defeated by an argument that is justified.

43

Argument-based Semantics

Example: Consider three arguments A, B and C



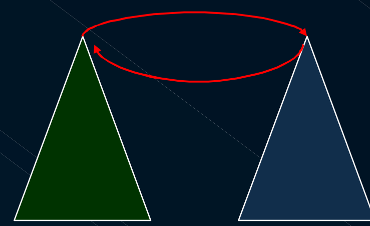
Argument A and C are justified; argument B is not.

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 44

Example: Even cycle

A = *"Nixon was a pacifist because he was a quaker"*

B = *"Nixon wasn't a pacifist because he was a republican"*



There are two status assignment that satisfy Def 1

Def. 1: Arguments are either *justified* or *not justified*

1. An argument is justified if all arguments defeating it (if any) are not justified.
2. An argument is not justified if it is defeated by an argument that is justified.

45

Argument-based Semantics

In the literature, two approaches to the solution of this problem can be found.

- *First approach:* changing Def. 1 in such a way that there is always precisely one possible way to assign a status to arguments. Undecided conflicts get the status 'not justified'.

  Allowing unique-status assignment (u.s.a).

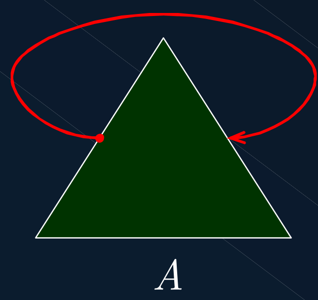
- *Second approach:* allowing multiple assignments, defining an argument as 'genuinely' justified iff it is justified in all possible assignments.

  Allowing multiple-status assignment (m.s.a).

Self-defeating Argument

Another problem with Definition 1

- The role of self-defeating arguments.



Self-defeating arguments are inconsistent with Definition 1

⇓ *but...*

They can be considered as plausible constructions.

The Unique-Status-Assignment Approach

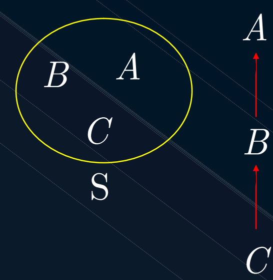
This idea could be presented in two different ways:

- Using a fixed-point operator
- Given a recursive definition of justified argument

Fixed-point Definitions

This approach has been used in several frameworks, *e.g.,* Pollock (1987,1992), Simari & Loui (1992) and Prakken & Sartor (1997). It is based on the notion of reinstatement, captured by Dung's definition of acceptability:

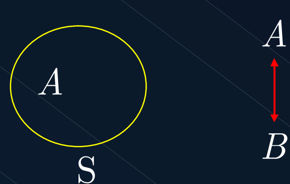
> Def. 2: (Acceptability)
> An argument A is acceptable wrt a set S of arguments iff each argument defeating A is defeated by an argument in S.



Computational Models for Argumentation in Multiagent Systems – EASSS 2005 49

A Fixed-point Operator

However, this notion seems to be not sufficient...



If $\mathrm{S}=\{A\}$, A is acceptable wrt S

> Def. 3: (Dung's Grounded Semantics) Let $Args$ be a set of arguments ordered by a binary relation of defeat, and let $\mathrm{S} \subseteq Args$. Then the operator F is defined as follows.
>
> $\mathrm{F}(\mathrm{S}) = \{\ A \in Args \mid A$ is acceptable wrt **S** $\}$

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 50

A Fixed-point Operator

Dung proves that the operator F has a least fixed point

Def. 4: (Justified Argument) An arg. is justified iff it is a member of the least fixed point of F.

Def. 5: (Least fixed point of F)

– $\mathrm{F}^0 = \varnothing$

– $\mathrm{F}^{i+1} = \{\ A \in Args \mid A \text{ is acceptable wrt } \mathrm{F}^i \}$

Propositions

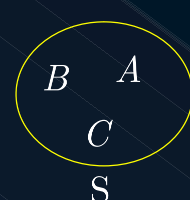
1. *All arguments in $\cup_{i=0..\infty} (\mathrm{F}^i)$ are justified.*
2. *If each argument is defeated by at most a finite number of arguments, then an argument is justified iff it is in $\cup_{i=0..\infty} (\ \mathrm{F}^i)$.*

Consider the previous example :

$\mathrm{F}^1 = \mathrm{F}(\varnothing) = \{C\}$

$\mathrm{F}^2 = \mathrm{F}(\mathrm{F}(\varnothing)) = \{A,\ C\}$

$\mathrm{F}^3 = \mathrm{F}(\mathrm{F}^2(\varnothing)) = \mathrm{F}^2$



G operator. Levels in Justification

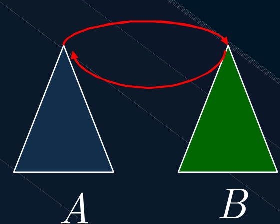
Def. 6: (G operator) Let $Args$ be a set of arguments ordered by a binary relation of defeat. Then

$\mathrm{G(S)}=\{A\in Args \mid A$ is not defeated by any arg. in $\mathrm{S}\}$

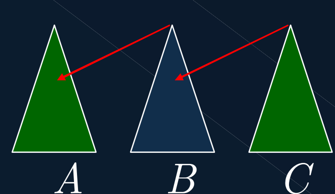
Def. 7: (Levels in justification)

- All arguments are *in* level 0
- An argument is *in* at level $(n+1)$ iff it is not defeated by any argument at level n
- An argument is *justified* iff there is an m such that for every $n \geq m$, the argument is in at level n.

Examples



Level	IN
0	A, B
1	
2	A, B
3	
4	A, B



Level	IN
0	A, B, C
1	C
2	A, C
3	A, C
4	$\ldots$

Infinite defeat chain

Consider an infinite chain of args $A_1, \ldots, A_n$ such that A_1 is defeated by A_2, A_2 is defeated by A_3, and so on.

$$A_1 \leftarrow A_2 \leftarrow A_3 \leftarrow \ldots$$

The least fixed point of this chain is empty, since no argument is undefeated. Consequently, $\mathrm{F}(\varnothing) = \varnothing$

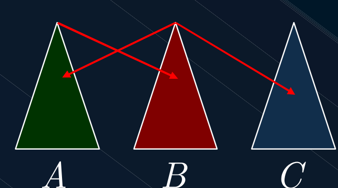
This example has two other fixed points:

$\mathrm{F}_1 = \{A_1, A_3, A_5, A_{7,} \ldots \}$

$\mathrm{F}_2 = \{A_2, A_4, A_6, A_{8,} \ldots\}$

Defensible and Overruled Arguments

Consider the following situation:

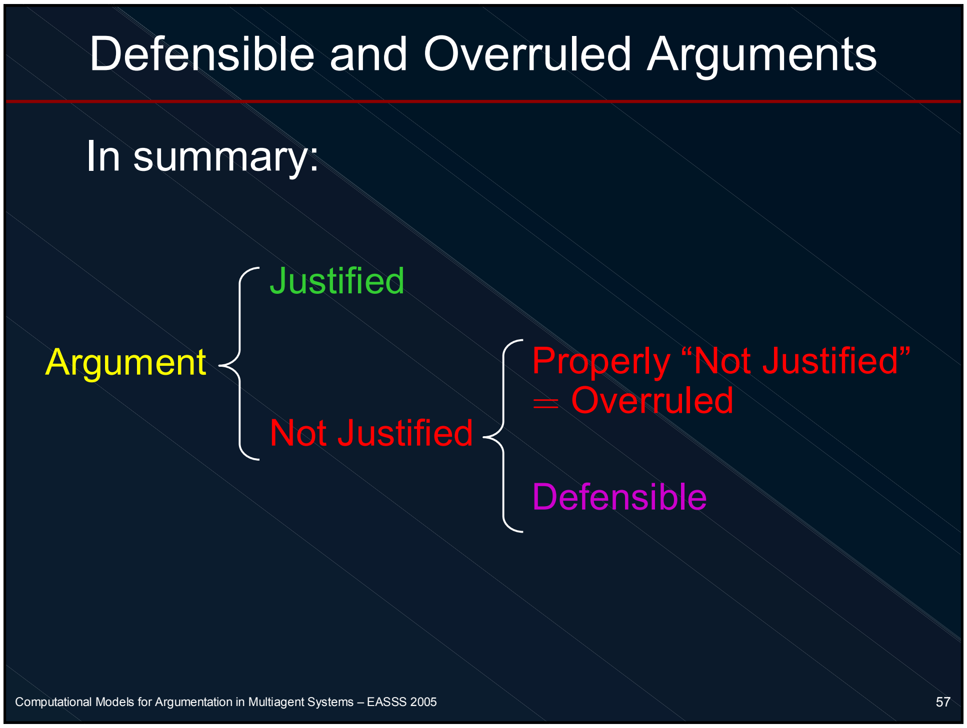


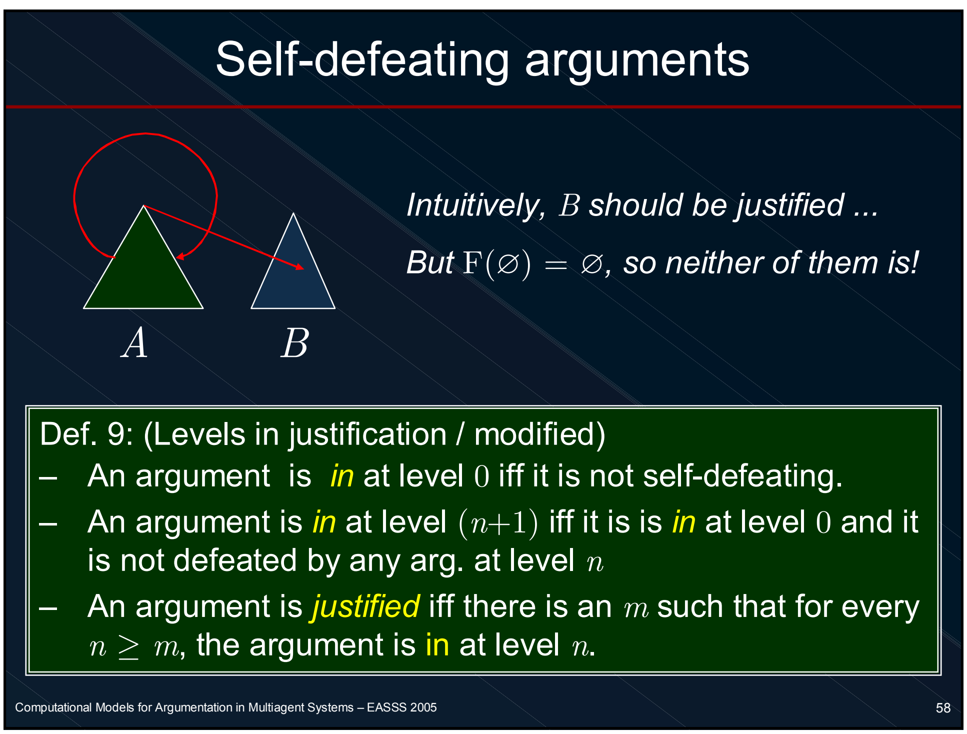
B is not defeated by a justified argument!

"B" is called "zombie argument" (Makinson & Schlechta,1991), or "defensible arguments" (Prakken & Sartor).

Def 8: (Overruled and defensible arguments)

- A is overruled iff A is not justified, and A is defeated by a justified argument
- A is defensible iff A is not justified and A is not overruled.


Defensible and Overruled Arguments
In summary:
Argument
Justified
Not Justified
Properly "Not Justified" = Overruled
Defensible
Computational Models for Argumentation in Multiagent Systems – EASSS 2005
57


Self-defeating arguments
A
B
Intuitively, B should be justified ...
But $\mathrm{F}(\varnothing) = \varnothing$, so neither of them is!
Def. 9: (Levels in justification / modified)
– An argument is *in* at level 0 iff it is not self-defeating.
– An argument is *in* at level $(n+1)$ iff it is is *in* at level 0 and it is not defeated by any arg. at level n
– An argument is *justified* iff there is an m such that for every $n \geq m$, the argument is in at level n.
Computational Models for Argumentation in Multiagent Systems – EASSS 2005
58

Self-defeating Arguments

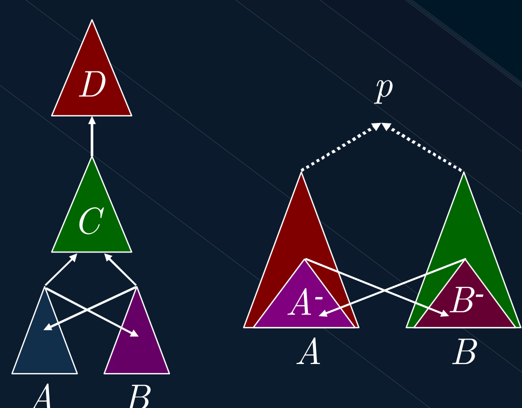
Appart from Pollock's refined version of "level-n arguments", there are other possible solutions to self-defeating arguments:

- Distinguishing a special empty argument which defeats any self-defeating argument (Prakken & Sartor, Vreeswijk).
- Demanding that by construction arguments must be non self-defeating, (Simari & Loui).

Problems with Unique-Status Assignment

There are some problems when evaluating unique-status assignment.

Example: Floating Arguments / Floating Conclusions



The unique-status approach is inherently unable to capture floating arguments and conclusions.

Using Multiple-Status Assignment

- A second way to deal with competing arguments of equal strenght is to let them induce two alternative *status assignments.*
- Evaluating outcomes from alternative status assignments let us determine when an argument is justified.

Def. : (Status assignment) Given a set S of args ordered by a binary defeat relation, an status assignment $sa(\mathrm{S})$ is a function which maps every argument in S into $\{in, out\}$, such that:

i. A is in iff all args defeating it (if any) are out.

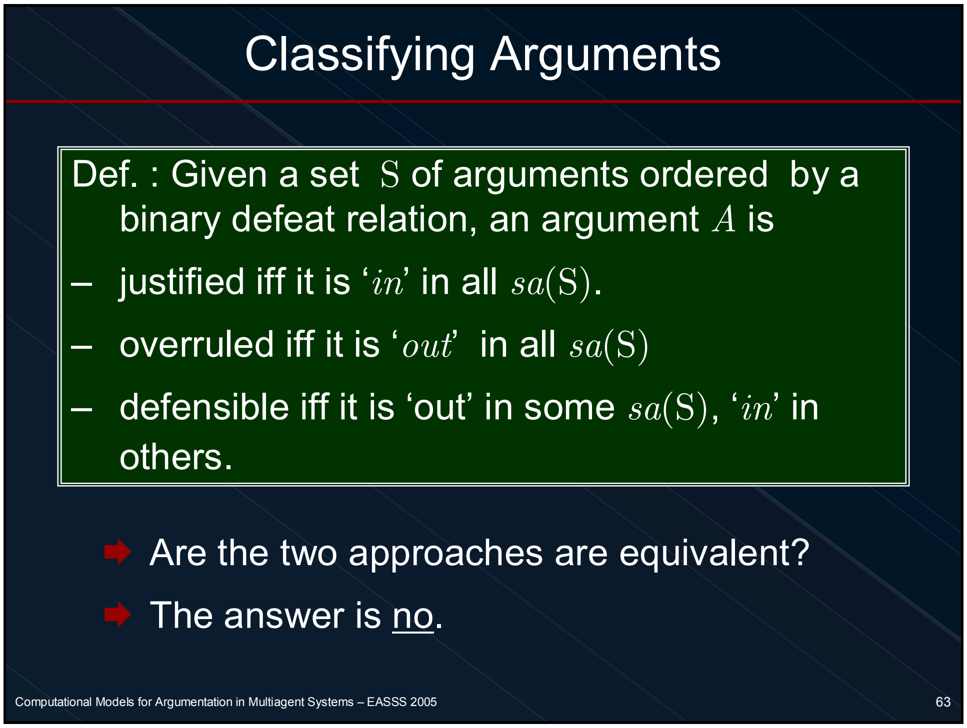
ii. A is out if it is defeated by an arg that is in.

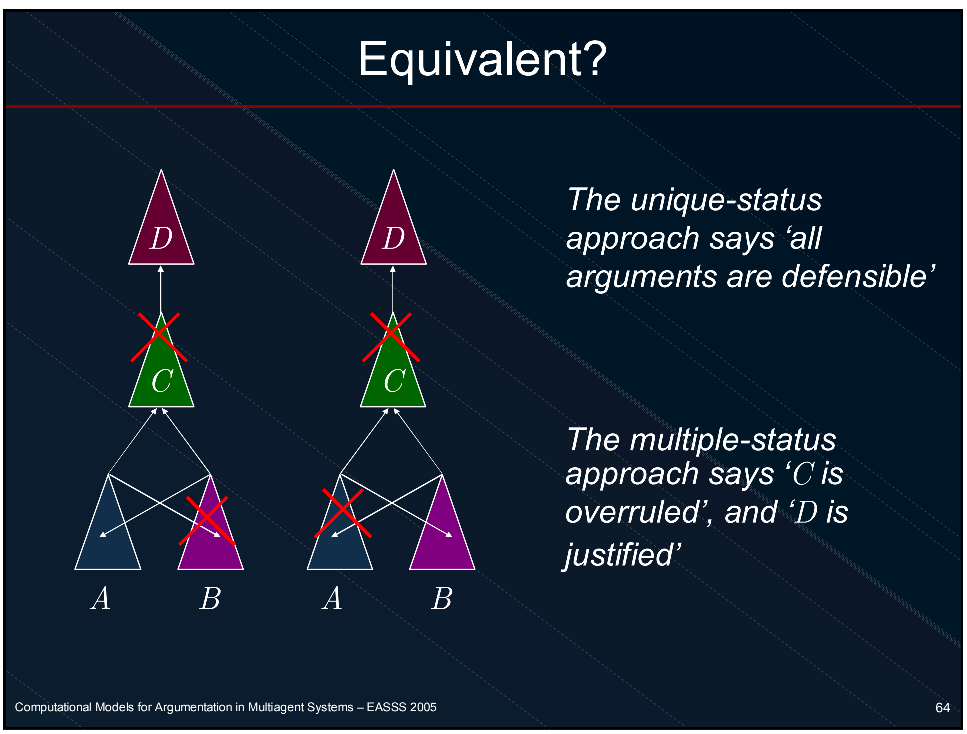
61

Example

Def. : (Justification) Given a set S of arguments ordered by a binary defeat relation, an argument is justified iff it is in in all possible status assignments to S.

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 62


Classifying Arguments
Def. : Given a set S of arguments ordered by a binary defeat relation, an argument A is
– justified iff it is 'in' in all $sa(\mathrm{S})$.
– overruled iff it is 'out' in all $sa(\mathrm{S})$
– defensible iff it is 'out' in some $sa(\mathrm{S})$, 'in' in others.
Are the two approaches are equivalent?
The answer is no.
Computational Models for Argumentation in Multiagent Systems – EASSS 2005
63


Equivalent?
D
C
A
B
D
C
A
B
The unique-status approach says 'all arguments are defensible'
The multiple-status approach says 'C is overruled', and 'D is justified'
Computational Models for Argumentation in Multiagent Systems – EASSS 2005
64

Status of Conclusions

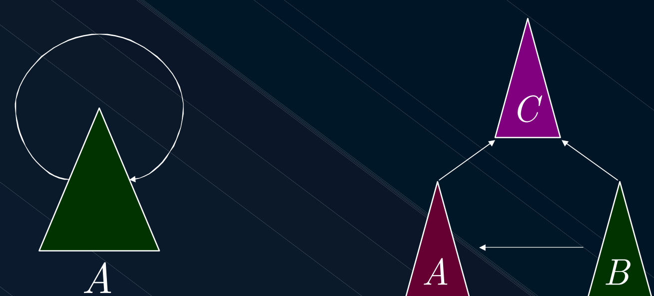
Def.: (Status of Conclusions)

- φ is a justified conclusion iff every status assignment assigns 'in' to an arg. with conclusion φ.
- φ is a defensible conclusion iff φ is not justified, and a conclusion of a defensible argument.
- φ is an overruled conclusion iff φ is not justified or defensible, and a conclusion of an overruled argument.

- Changing the first clause into 'φ is a justified conclusion iff φ is the conclusion of a justified argument' would make a stronger notion ...

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 65

Problems with Multiple-Status Assignment



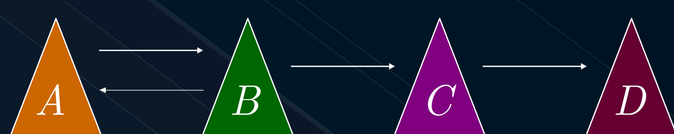
- What are the status assignments?
- There are no status assignments!

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 66

Comparing the two approaches

- Some researchers say that the difference between the two approaches can be compared with the ‘skeptical’ vs. ‘credulous’ attitude towards drawing defeasible conclusions ...
- m.s.a is more convenient for identifying sets of arguments that are compatible with each other.
- u.s.a considers arguments on an individual basis.

Example



Note that A and D are somehow incompatible; in the unique-assignment approach this notion is (or seems) harder to capture.

- This example has 2 status assignments: $\{A,\ C\}$ and $\{B,\ D\}$

Outline

- (Very brief) Introduction to Multiagent Systems
- What is argumentation? Fundamentals
- A Case Study: DeLP and its extensions as an argument-based approach to logic programming.
- Argument-based negotiation
- Conclusions

Deafeasible Logic Programming: DeLP

A *Defeasible Logic Program* (*dlp*) is a set of facts, strict and defeasible rules denoted $\mathcal{P} = (\Pi, \Delta)$

Π Strict Rules:

$$bird(X) \leftarrow chicken(X)$$
$$bird\ (X) \leftarrow penguin(X)$$
$$\neg flies(X) \leftarrow penguin(X)$$

Facts:

$$chicken(tina)$$
$$penguin(opus)$$
$$scared(tina)$$

Δ Defeasible Rules:

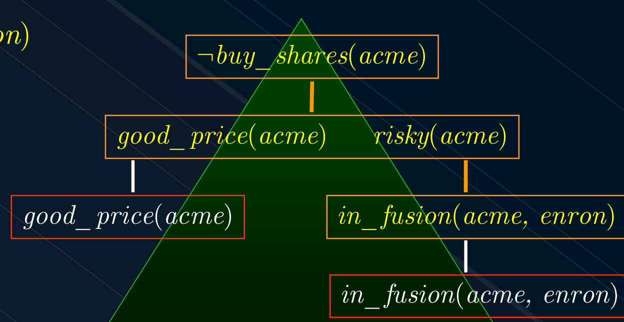
$$flies(X) \prec bird(X)$$
$$\neg flies(X) \prec chicken(X)$$
$$flies(X) \prec chicken\ (X),\ scared(X)$$

Defeasible Argumentation

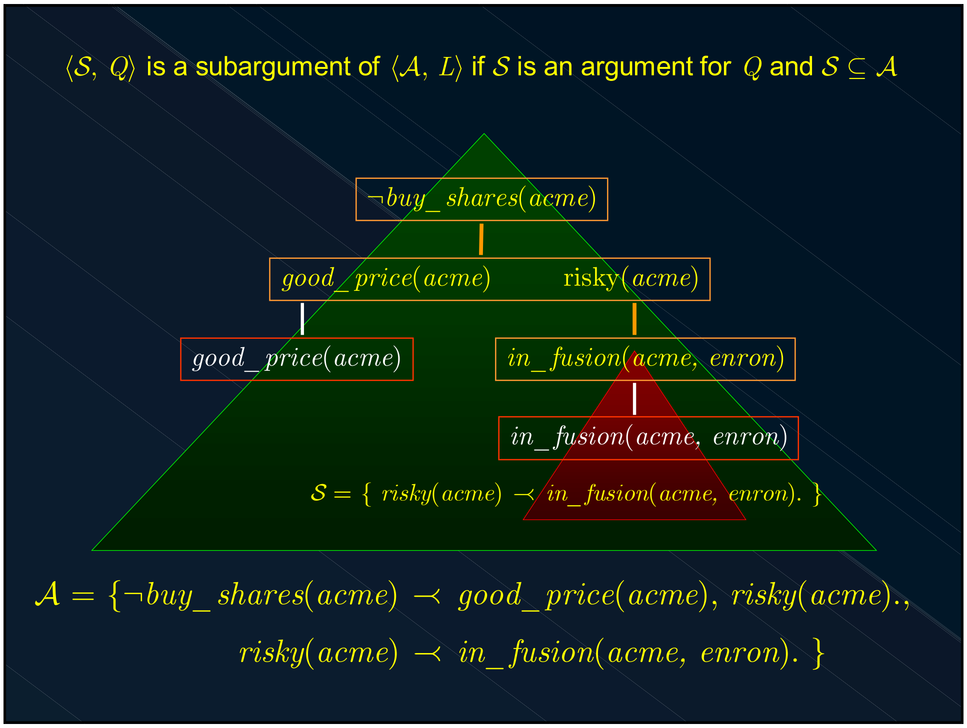
Def: Let L be a literal and $\mathcal{P} = (\Pi, \Delta)$ be a program. $\langle \mathcal{A}, L \rangle$ is an *argument*, for L, if $\mathcal{A}$ is a set of rules in Δ such that:

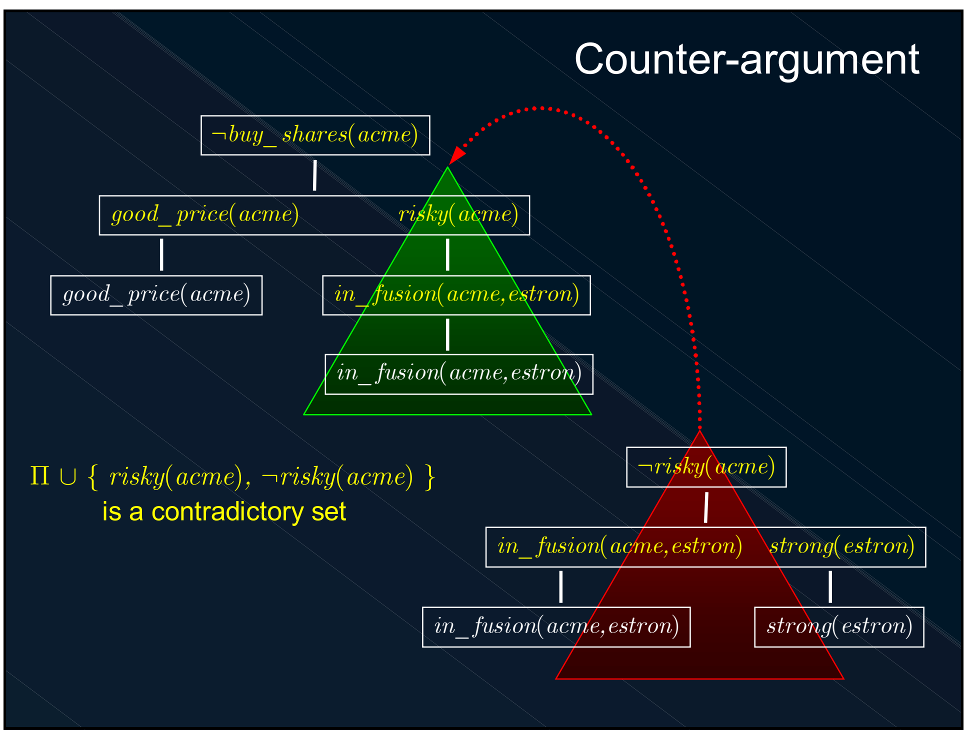
1) There exists a defeasible derivation of L from $\Pi \cup \mathcal{A}$;
2) The set $\Pi \cup \mathcal{A}$ is non contradictory; and
3) There is no proper subset $\mathcal{A}'$ of $\mathcal{A}$ such that $\mathcal{A}'$ satisfies 1) and 2).

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



$\langle \{\neg buy\_shares(acme) \prec good\_price(acme), risky(acme).,$
$risky(acme) \prec in\_fusion(acme, enron).\}, \neg buy\_shares(acme) \rangle$


⟨𝒮, Q⟩ is a subargument of ⟨𝒜, L⟩ if 𝒮 is an argument for Q and 𝒮 ⊆ 𝒜
¬buy_shares(acme)
good_price(acme) risky(acme)
good_price(acme)
in_fusion(acme, enron)
in_fusion(acme, enron)
𝒮 = { risky(acme) ⤙ in_fusion(acme, enron). }
𝒜 = {¬buy_shares(acme) ⤙ good_price(acme), risky(acme).,
risky(acme) ⤙ in_fusion(acme, enron). }


Counter-argument
¬buy_shares(acme)
good_price(acme) risky(acme)
good_price(acme)
in_fusion(acme,estron)
in_fusion(acme,estron)
Π ∪ { risky(acme), ¬risky(acme) }
is a contradictory set
¬risky(acme)
in_fusion(acme,estron) strong(estron)
in_fusion(acme,estron)
strong(estron)

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.)

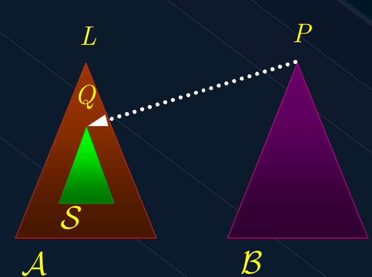
75

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 $\langle \mathcal{A}, L \rangle$ that atacks 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

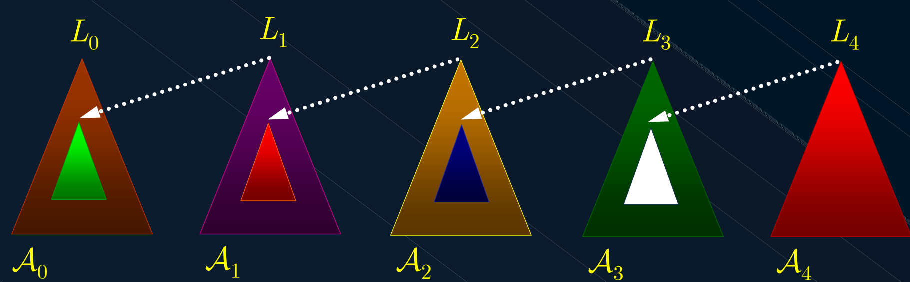
(b) $\langle \mathcal{B}, P \rangle$ *is not comparable to* $\langle \mathcal{S}, Q \rangle$ (*blocking defeater*)



Computational Models for Argumentation in Multiagent Systems – EASSS 2005 76

Argumentation Line

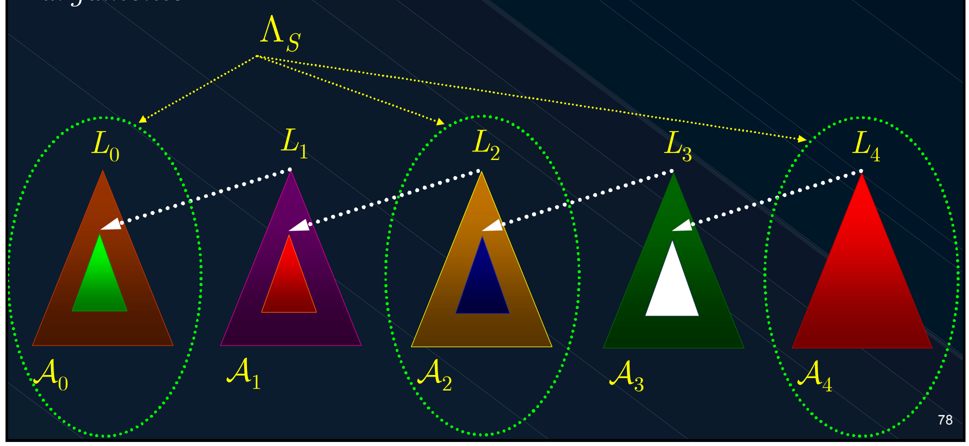
Given $\mathcal{P} = (\Pi, \Delta)$, and $\langle \mathcal{A}_0, L_0 \rangle$ an argument obtained from $\mathcal{P}$. An *argumentation line* for $\langle \mathcal{A}_0, L_0 \rangle$ is a sequence of arguments obtained from $\mathcal{P}$, denoted $\Lambda = [\langle \mathcal{A}_0, L_0 \rangle, \langle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



77

Argumentation Line

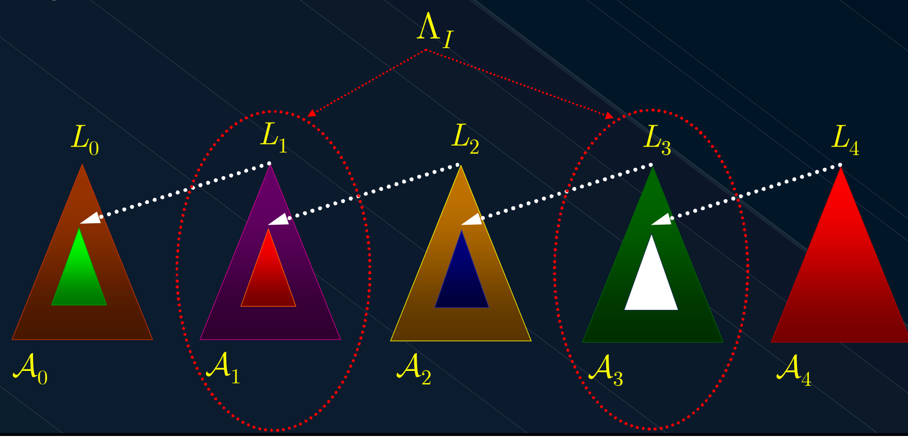
Given an argumentation line $\Lambda = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \ldots]$, the subsequence $\Lambda_S = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_2, L_2 \rangle, \ldots]$ contains *supporting arguments* and $\Lambda_I = [\langle \mathcal{A}_1,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



78

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*.



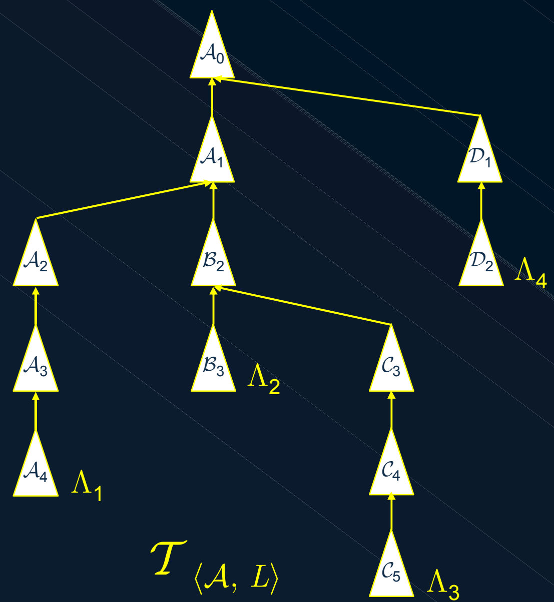
79

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.
2. The sets 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).

Computational Models for Argumentation in Multiagent Systems – EASSS 2005 80



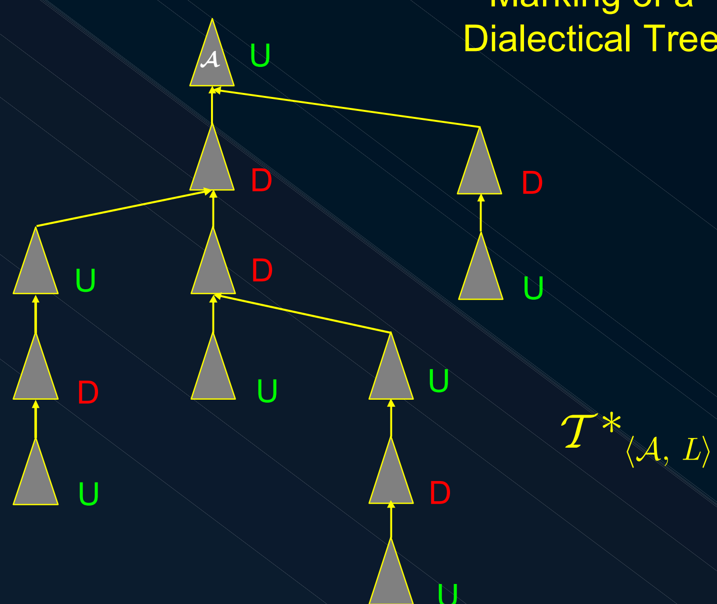
Dialectical Tree

Given a program $\mathcal{P} = (\Pi, \Delta)$, a literal L will be *warranted* if there is an argument $\langle \mathcal{A}, L \rangle$ built from $\mathcal{P}$, and that argument has a dialectical tree whose root node is marked *U*.

That is, argument $\langle \mathcal{A}, L \rangle$ is an argument for which all the possible defeaters have been defeated.

We will say that *$\mathcal{A}$ is a warrant for L*.

Marking of a Dialectical Tree



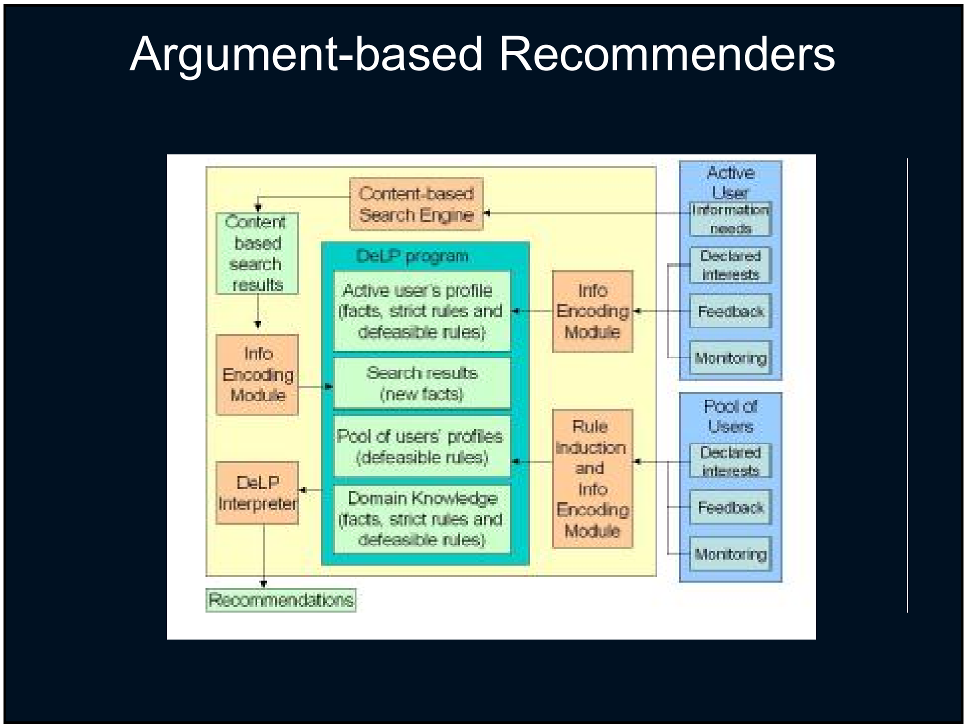
Answers in DeLP

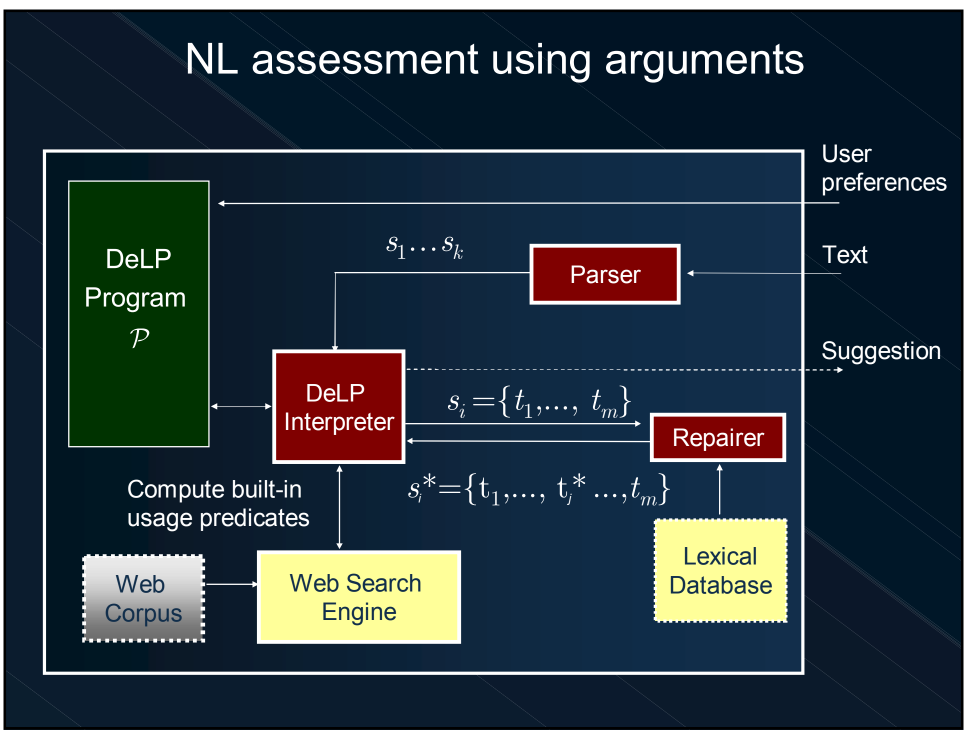
Given a program $\mathcal{P} = (\Pi, \Delta)$, and a query for L the posible answers are:

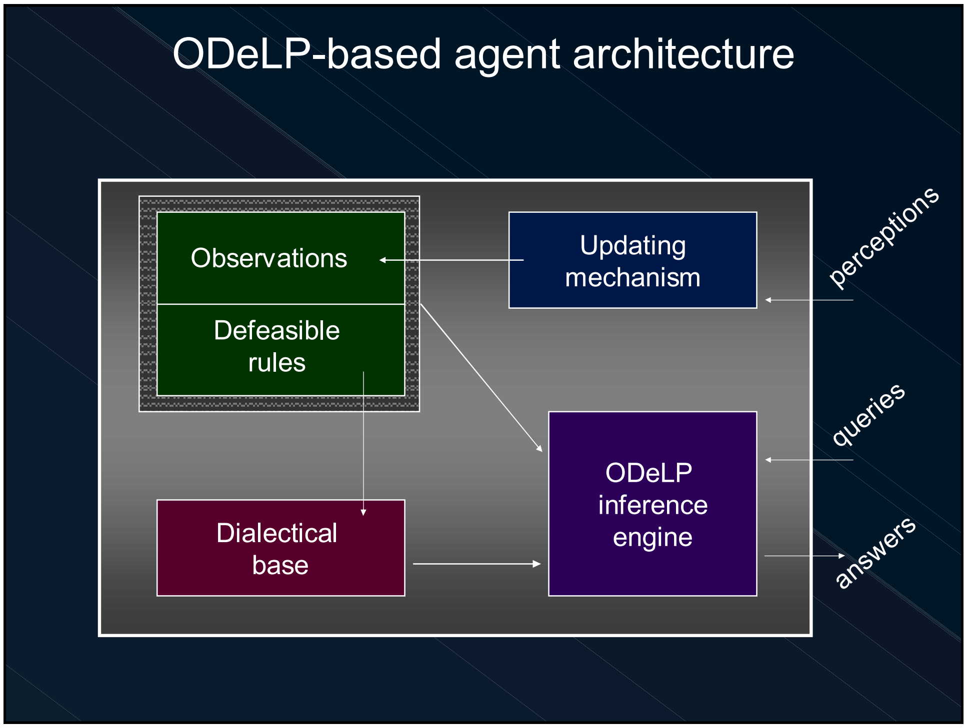
- *YES*, if L is warranted.
- *NO*, if $\neg L$ is warranted.
- *UNDECIDED*, if neither L nor $\neg L$ are warranted.
- *UNKNOWN,* if L is not in the language of the program.

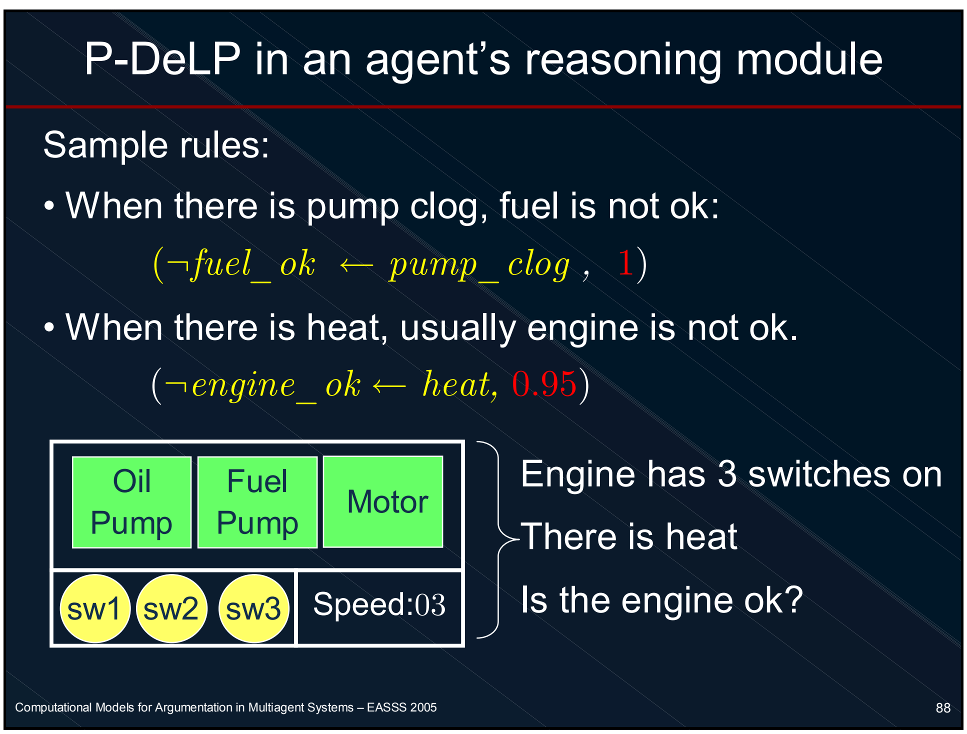
DeLP : extensions

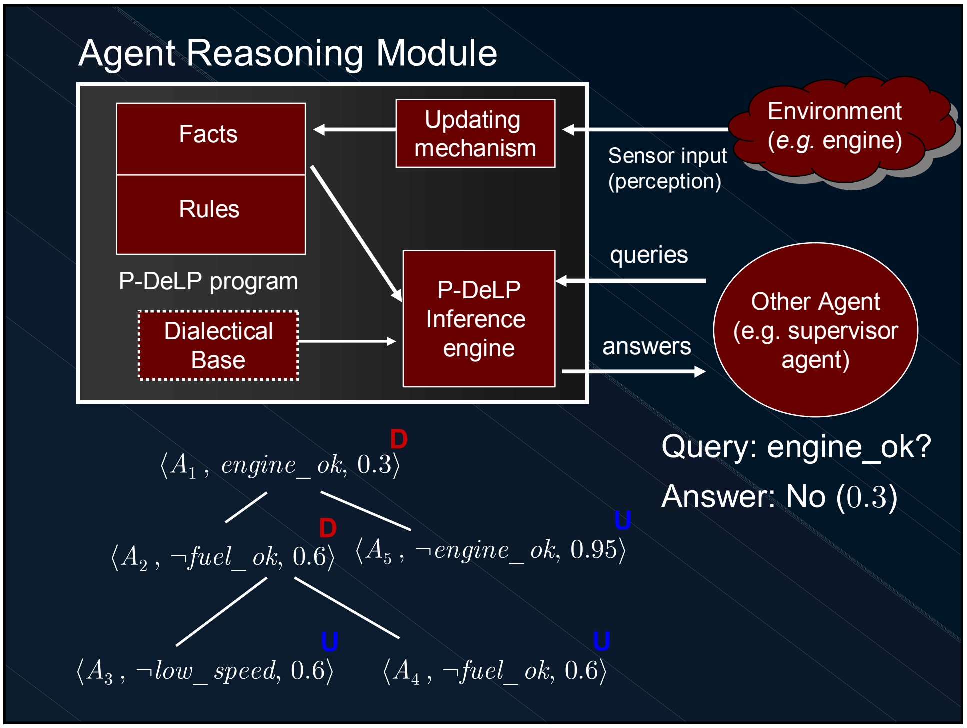
- *Recently extensions of DeLP have been developed:*
  - *P-DeLP (Chesñevar et. al, 2004): aims at modelling reasoning under uncertainty (e.g. possibilistic reasoning).*
  - *O-DeLP (Capobianco et. al, 2004): aims at modelling reasoning for agents in changing environments.*

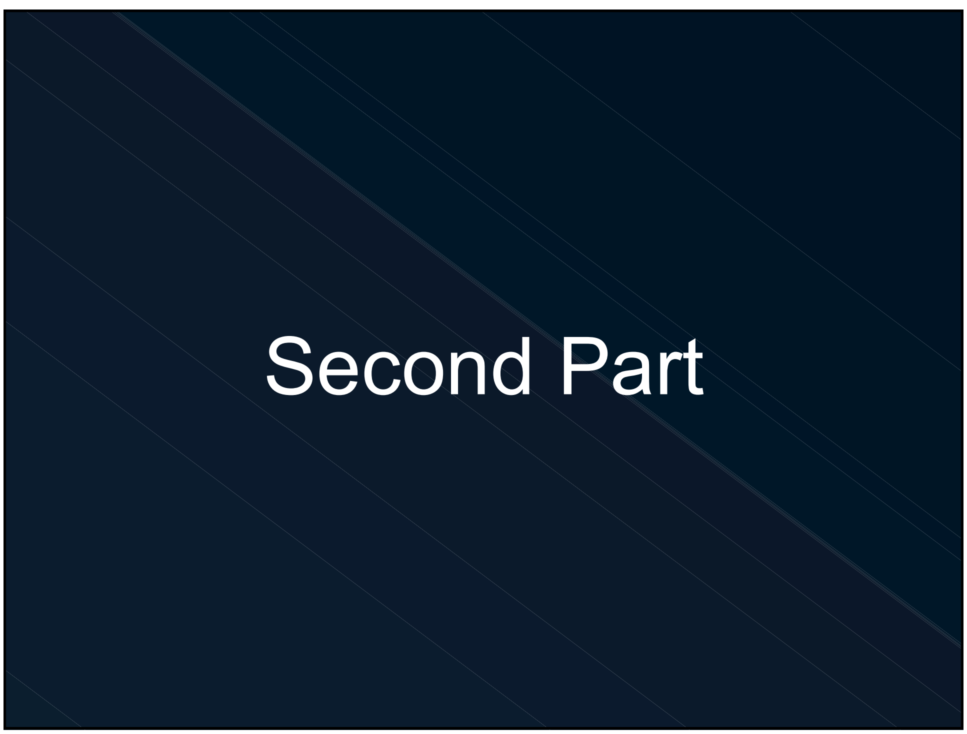

Argument-based Recommenders
Content-based Search Engine
Content based search results
DeLP program
Active user's profile (facts, strict rules and defeasible rules)
Search results (new facts)
Pool of users' profiles (defeasible rules)
Domain Knowledge (facts, strict rules and defeasible rules)
Info Encoding Module
Info Encoding Module
DeLP Interpreter
Rule Induction and Info Encoding Module
Active User
Information needs
Declared interests
Feedback
Monitoring
Pool of Users
Declared interests
Feedback
Monitoring
Recommendations


NL assessment using arguments
DeLP Program
$\mathcal{P}$
User preferences
$s_1 \ldots s_k$
Parser
Text
DeLP Interpreter
Suggestion
$s_i = \{t_1, \ldots, t_m\}$
Repairer
Compute built-in usage predicates
$s_i^* = \{t_1, \ldots, t_j^* \ldots, t_m\}$
Web Corpus
Web Search Engine
Lexical Database


ODeLP-based agent architecture
Observations
Defeasible rules
Updating mechanism
perceptions
queries
answers
Dialectical base
ODeLP inference engine


P-DeLP in an agent's reasoning module
Sample rules:
• When there is pump clog, fuel is not ok:
(¬fuel_ok ← pump_clog , 1)
• When there is heat, usually engine is not ok.
(¬engine_ok ← heat, 0.95)
Oil Pump
Fuel Pump
Motor
sw1
sw2
sw3
Speed:03
Engine has 3 switches on
There is heat
Is the engine ok?
Computational Models for Argumentation in Multiagent Systems – EASSS 2005
88


Agent Reasoning Module
Facts
Rules
P-DeLP program
Updating mechanism
Dialectical Base
P-DeLP Inference engine
Environment (e.g. engine)
Sensor input (perception)
queries
answers
Other Agent (e.g. supervisor agent)
D
$\langle A_1, engine\_ok, 0.3 \rangle$
D
$\langle A_2, \neg fuel\_ok, 0.6 \rangle$
U
$\langle A_5, \neg engine\_ok, 0.95 \rangle$
U
$\langle A_3, \neg low\_speed, 0.6 \rangle$
U
$\langle A_4, \neg fuel\_ok, 0.6 \rangle$
Query: engine_ok?
Answer: No (0.3)


Second Part