DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Structured Argumentation: DeLP

– *Guillermo R. Simari* (grs@cs.uns.edu.ar)

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (*LIDIA*)

Instituto de Ciencias e Ingeniería de la Computación

Departamento de Ciencias e Ingeniería de la Computación

UNIVERSIDAD NACIONAL DEL SUR

Bahia Blanca - ARGENTINA

ICIC CONICET UNS

Defeasible Logic Programming

Introduction

- *The inference engine is based in a defeasible argumentation inference mechanism for warranting the conclusions.*
- *In the language of DeLP there is the possibility of representing information in the form of weak rules in a declarative manner.*
- *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

A couple of premises:

- *General Common Sense reasoning should be defeasible in a way that is not explicitly programmed.*
- *Acceptance and rejection should result from 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.

Conceptual View

Let’s begin with the accepted view that there are five common elements in systems for defeasible argumentation:

Definition of Status of Arguments

Definition of Defeat among Arguments

Definition of Conflict among arguments

Definition of Argument

Definition of the Underlying (Logical) Language

Conceptual View

Definition of Status of Arguments
Definition of Defeat among Arguments
Definition of Conflict among arguments
Definition of Argument
Definition of the Underlying (Logical) Language

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

$$mammal(idefix) \leftarrow dog(idefix)$$

$$bird(opus) \leftarrow penguin(opus)$$

- 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.

Language: Facts and Strict and Defeasible 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:

$$\sim guilty(joe) \leftarrow innocent(joe)$$

$$mammal(garfield) \leftarrow cat(garfield)$$

Language: Facts and Strict and Defeasible Rules

- A *Defeasible Rule* is denoted: $L_0 \prec L_1, L_2, \ldots, L_n$
- 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 the new knowledge to the representation, 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 $\mathit{Defeasible\ Logic\ Program}$ (delp) denoted $\mathcal{P} = (\Pi, \Delta)$, where Π is a set of facts and strict rules, and Δ is a set of defeasible rules.

Π

Strict Rules

$$\left\{\begin{array}{l} bird(X) \leftarrow chicken(X) \\ bird\ (X) \leftarrow penguin(X) \\ {\sim}flies(X) \leftarrow penguin(X) \end{array}\right.$$

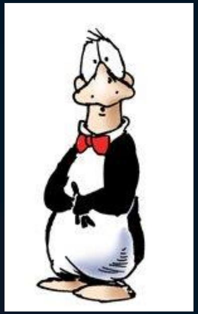
Facts

$$\left.\begin{array}{l} chicken(tina) \\ penguin(opus) \\ scared(tina) \end{array}\right\}$$

Δ

Defeasible Rules

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



Defeasible Logic Programming: DeLP

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

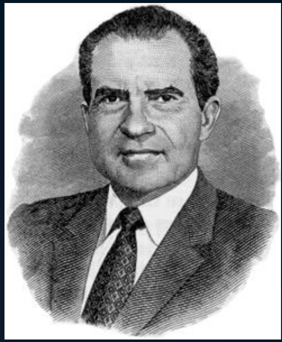
Δ

Defeasible Rules

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

Π

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



Adapted from Prakken and Vreeswijk (2000)

Defeasible Logic Programming: DeLP

Still one more 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

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

Conceptual View

Definition of Status of Arguments
Definition of Defeat among Arguments
Definition of Conflict among arguments
Definition of Argument
Definition of the Underlying (Logical) Language

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 $\mathcal{P} \mid\!\sim L$, is a finite sequence of ground literals

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

such that each literal L_k $(1 \leq i \leq n)$ 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 ($i < k$).

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 that contradicts L, or the way that L is inferred, that will prevent the acceptance of L as a valid conclusion.*
- *A few examples of defeasible derivation follow.*

Defeasible Derivation

From the program:

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

The following are some derivations that could be obtained:

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

i.e., $\mathcal{P} \mid\!\sim flies(tina)$, and $\mathcal{P} \mid\!\sim\ \sim flies(opus)$

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.

$poor\_perf(john).\quad sick(john).$

$good\_perf(peter).\quad unruly(peter).$

$suspend(X) \prec \sim responsible(X).$

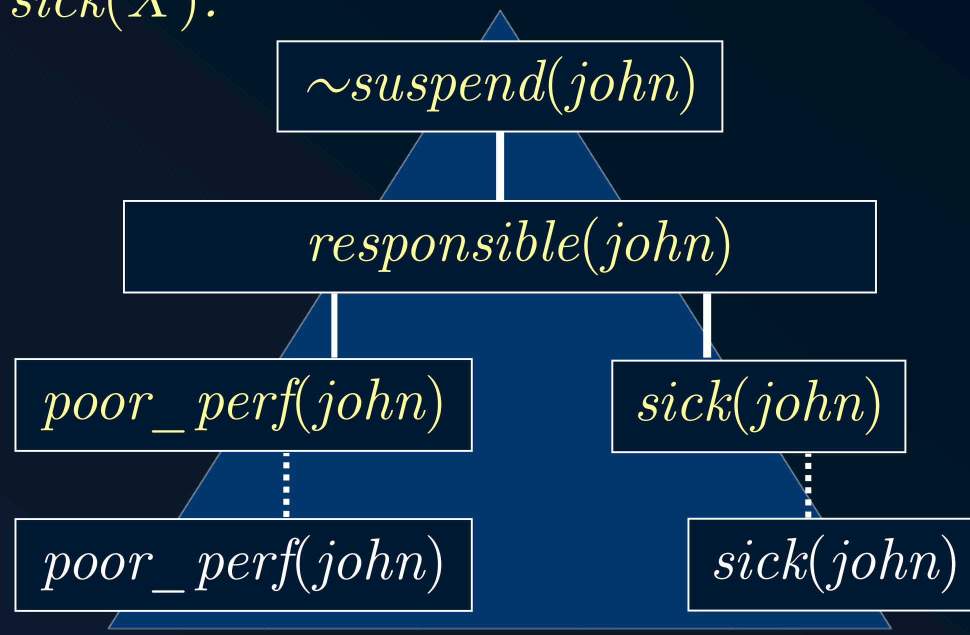
$suspend(X) \prec unruly(X).$

$\sim suspend(X) \prec responsible(X).$

$\sim responsible(X) \prec poor\_perf(X).$

$responsible(X) \prec good\_perf(X).$

$responsible(X) \prec poor\_perf(X),\ sick(X).$



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

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

$poor\_perf(john).\quad sick(john).$

$good\_perf(peter).\quad unruly(peter).$

$suspend(X) \mathrel{-\!\!\!\prec} {\sim}responsible(X).$

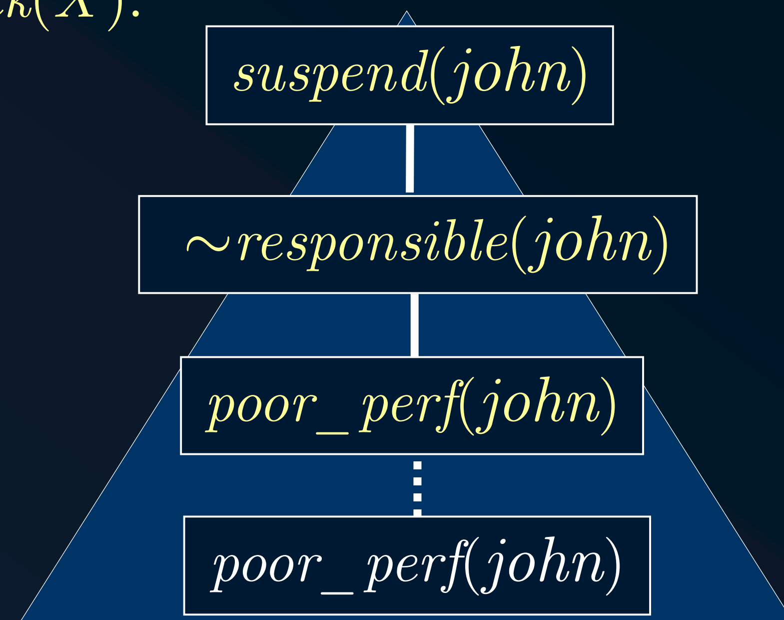
$suspend(X) \mathrel{-\!\!\!\prec} unruly(X).$

${\sim}suspend(X) \mathrel{-\!\!\!\prec} responsible(X).$

${\sim}responsible(X) \mathrel{-\!\!\!\prec} poor\_perf(X).$

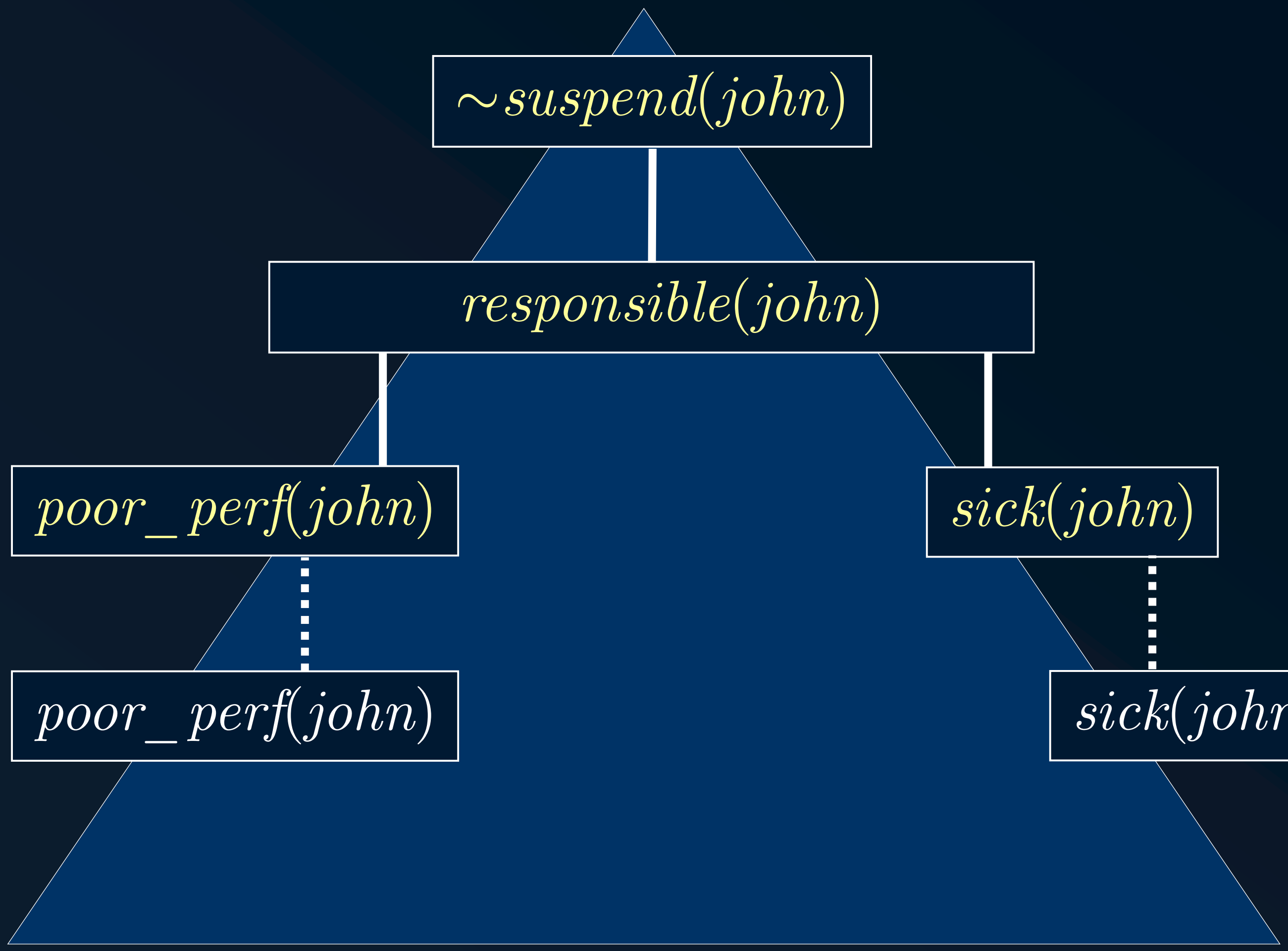
$responsible(X) \mathrel{-\!\!\!\prec} good\_perf(X).$

$responsible(X) \mathrel{-\!\!\!\prec} poor\_perf(X),\ sick(X).$



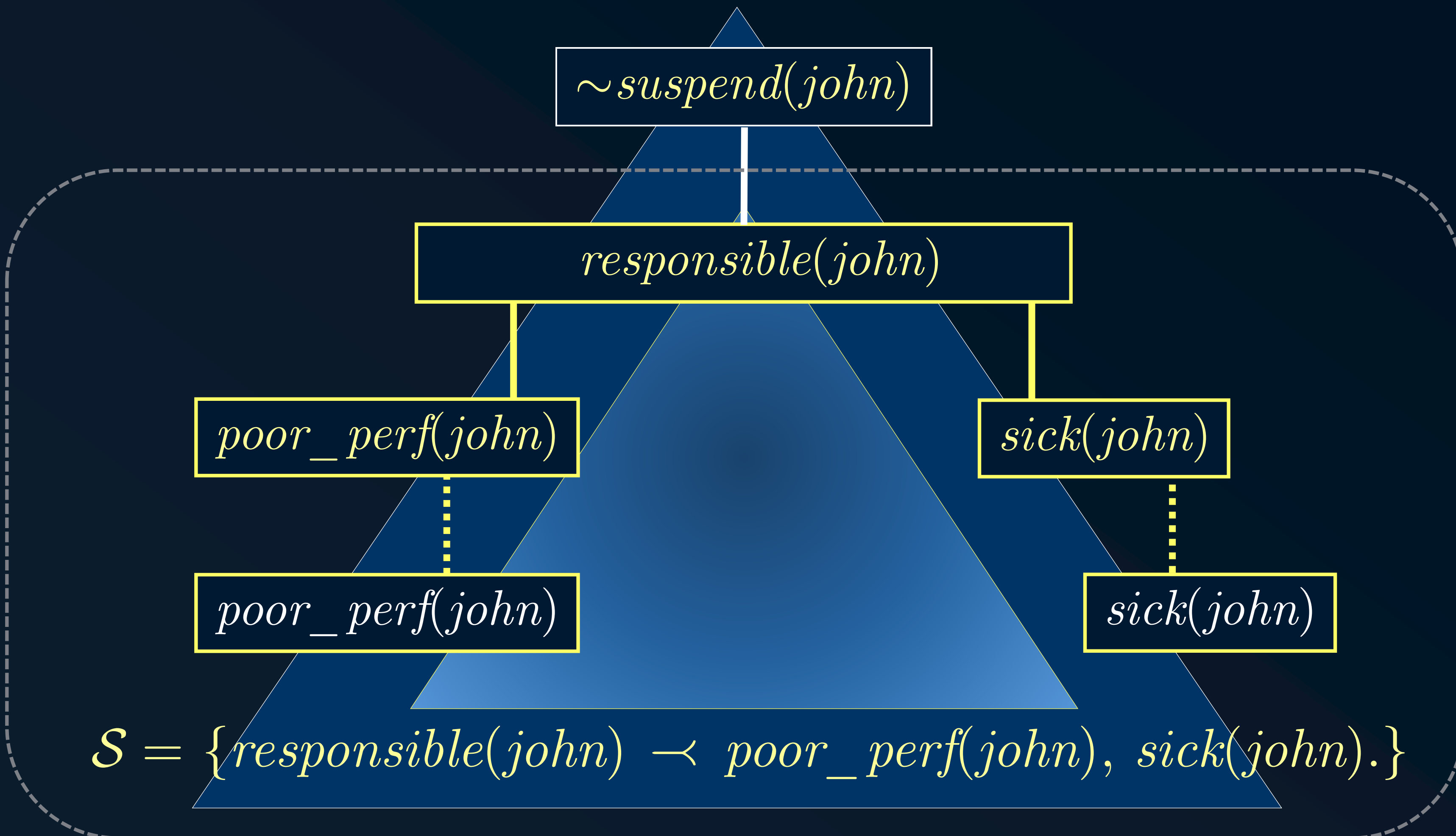
An argument for
$suspend(john)$
built from the program above

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



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

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



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

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

Conceptual View

Definition of Status of Arguments
Definition of Defeat among Arguments
Definition of Conflict among arguments
Definition of Argument
Definition of the Underlying (Logical) Language

Counter-Arguments or Rebuttals

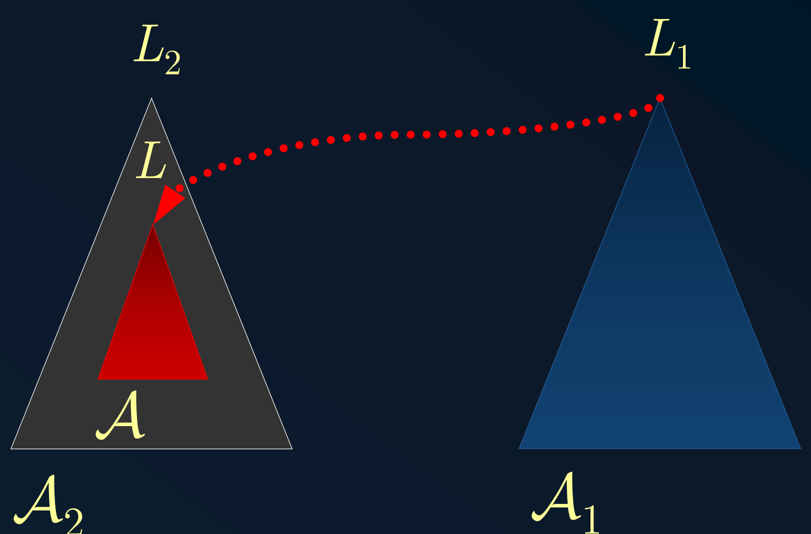
- In DeLP, answers are supported by arguments but an argument could be defeated by other arguments.
- Informally, a query L will succeed if the supporting argument for it is not defeated.
- In order to study this situation, counter-arguments or rebuttals are considered.
- Counter-arguments are also arguments,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

Counter-Arguments or Rebuttals

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.

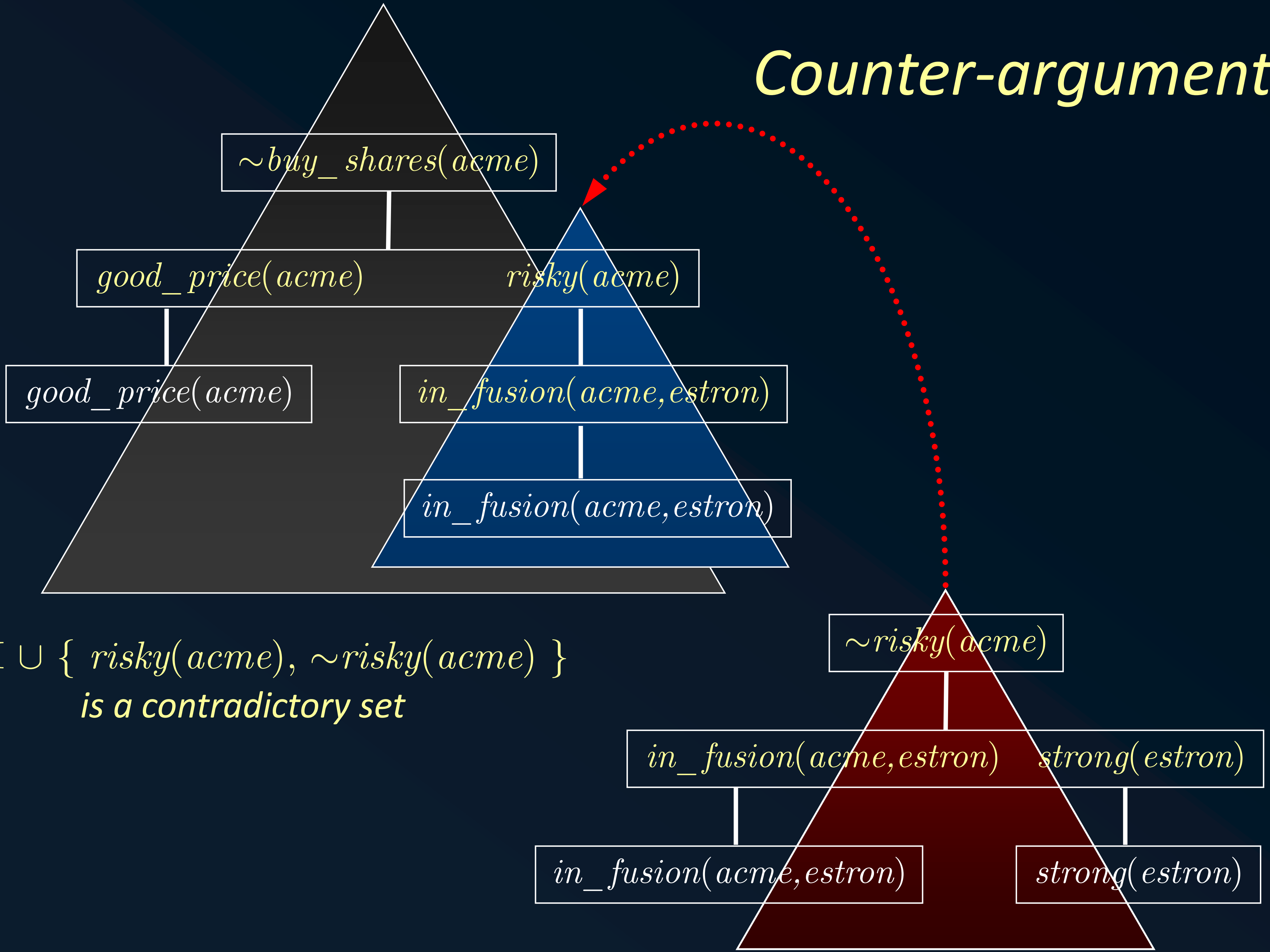
Def: Let $\mathcal{P} = (\Pi, \Delta)$ be a program. We say that $\langle \mathcal{A}_1,\ L_1 \rangle$ *counter-argues*, *rebuts* or *attacks* $\langle \mathcal{A}_2,\ L_2 \rangle$ at literal L, if and only if there exists a sub-argument $\langle \mathcal{A},\ L \rangle$ of $\langle \mathcal{A}_2,\ L_2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



Counter-Arguments or Rebuttals

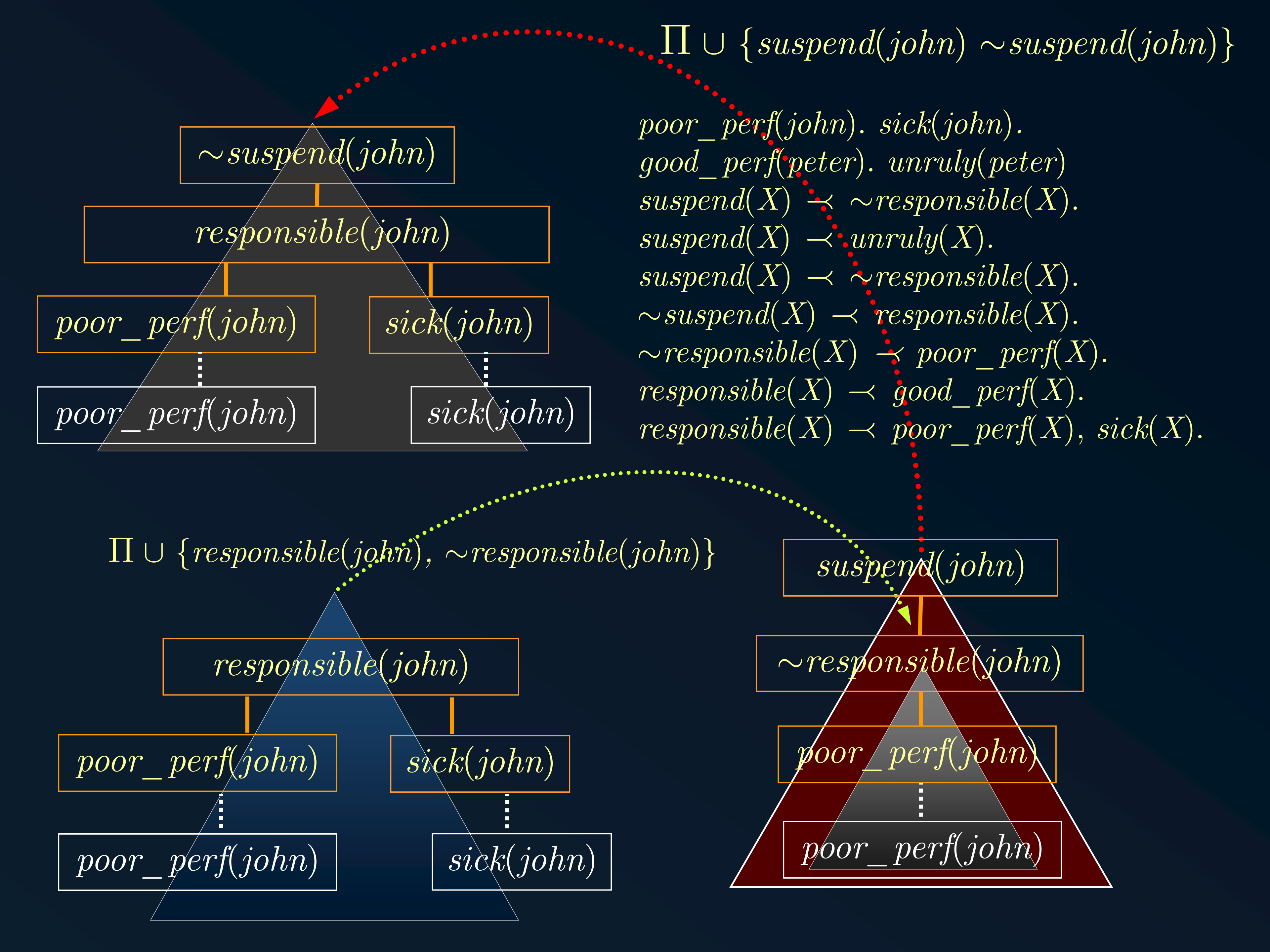
- Given $\mathcal{P} = (\Pi, \Delta)$, any literal P such that $\Pi \vdash P$, has the support of the empty argument $\langle \varnothing, P \rangle$.
- Clearly, there is no posible counter-argument for any of those P since there is no way of constructing an argument which would mention a literal in 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



$\Pi \cup \{\ risky(acme),\ \sim risky(acme)\ \}$

is a contradictory set

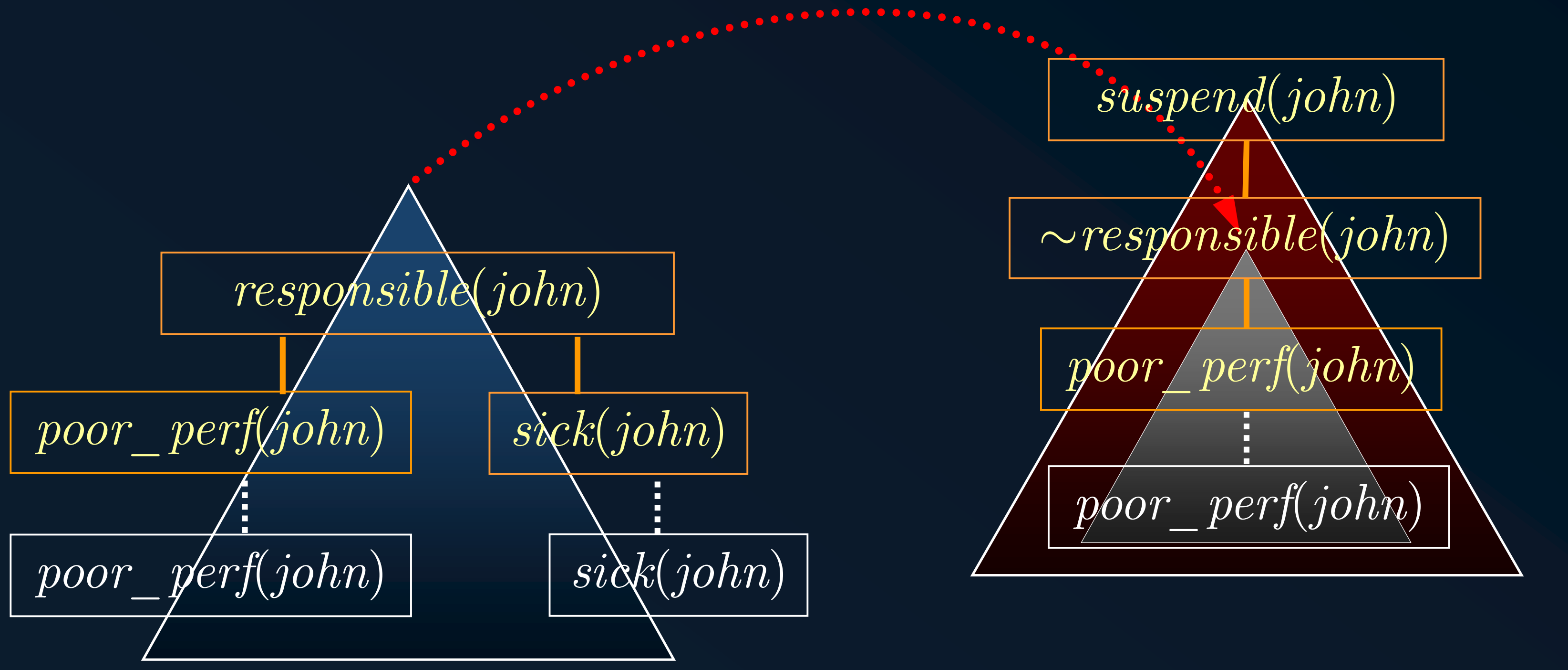

Π ∪ {suspend(john) ∼suspend(john)}
poor_perf(john). sick(john).
good_perf(peter). unruly(peter)
suspend(X) ≺ ∼responsible(X).
suspend(X) ≺ unruly(X).
suspend(X) ≺ ∼responsible(X).
∼suspend(X) ≺ responsible(X).
∼responsible(X) ≺ poor_perf(X).
responsible(X) ≺ good_perf(X).
responsible(X) ≺ poor_perf(X), sick(X).
∼suspend(john)
responsible(john)
poor_perf(john)
sick(john)
poor_perf(john)
sick(john)
Π ∪ {responsible(john), ∼responsible(john)}
responsible(john)
poor_perf(john)
sick(john)
poor_perf(john)
sick(john)
suspend(john)
∼responsible(john)
poor_perf(john)
poor_perf(john)

Conceptual View

Definition of Status of Arguments
Definition of Defeat among Arguments
Definition of Conflict among arguments
Definition of Argument
Definition of the Underlying (Logical) Language

Defeaters

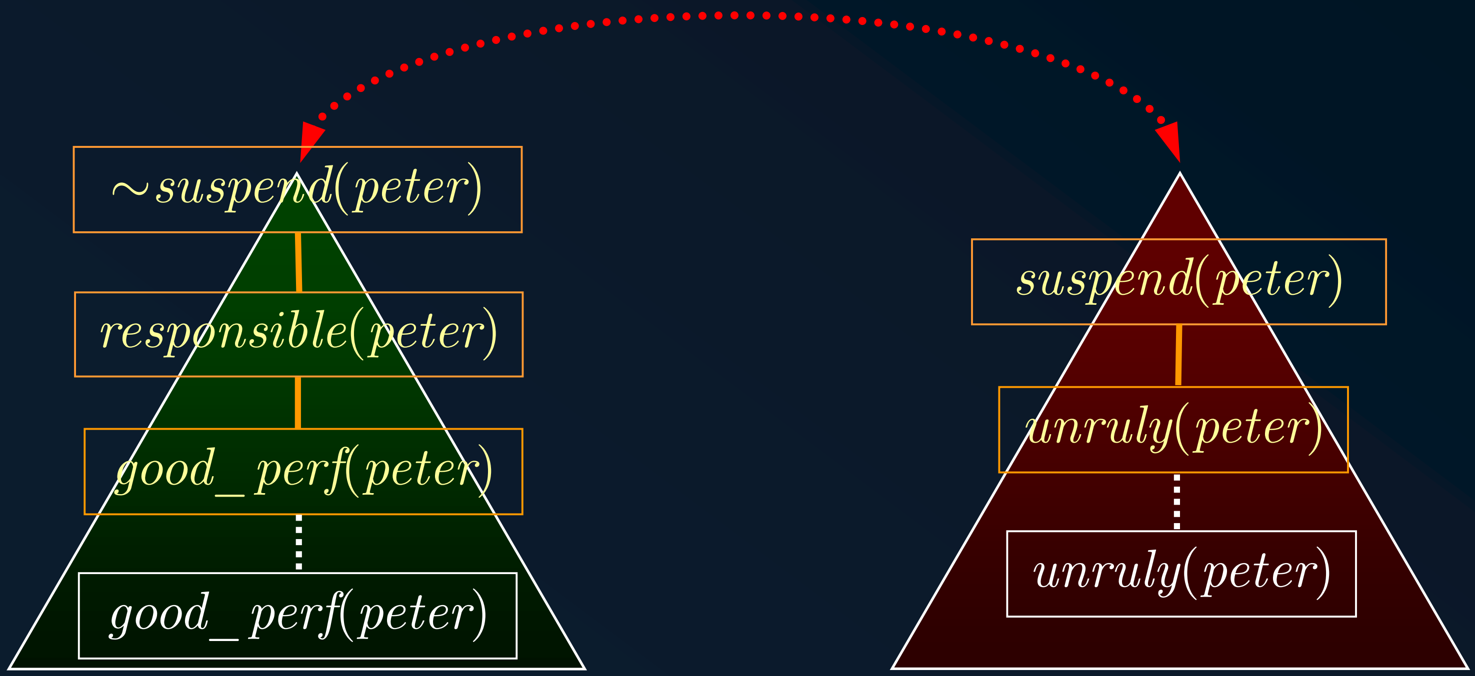
An argument $\langle \mathcal{B}, P\rangle$ is a $proper\ defeater$ for $\langle \mathcal{A}, L\rangle$ if $\langle \mathcal{B}, P\rangle$ is a counter-argument of $\langle \mathcal{A}, L\rangle$ that attacks a subargument $\langle \mathcal{S}, Q\rangle$ of $\langle \mathcal{A}, L\rangle$ and $\langle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



Proper Defeater

Defeaters

An argument $\langle \mathcal{B},\ P \rangle$ is a $blocking\ defeater$ 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 $\Pi_G \cup \mathcal{H} \cup \mathcal{A}_1 \mid\!\sim L_1$ while $\Pi_G \cup \mathcal{H} \not\mid\!\sim L_1$ then $\Pi_G \cup \mathcal{H} \cup \mathcal{A}_1 \mid\!\sim L_2$, and
2. There exists $\mathcal{H}' \subseteq \mathcal{F}$ such that there exists a defeasible derivation $\Pi_G \cup \mathcal{H}' \cup \mathcal{A}_2 \mid\!\sim L_2$ and $\Pi_G \cup \mathcal{H}' \not\mid\!\sim L_2$ but $\Pi_G \cup \mathcal{H}' \cup \mathcal{A}_1 \not\mid\!\sim L_1$

Intuitive view of Specificity - DeLP (Comparison)

More informed arguments are preferred over less informed ones:

$$\langle \{ \sim a \prec b \}, \sim a \rangle \preceq \langle \{ a \prec b, c \}, a \rangle$$

Shorter derivations are preferred over longer derivations:

$$\langle \{ (\sim a \prec b); (b \prec c) \}, \sim a \rangle \preceq \langle \{ a \prec b \}, a \rangle$$

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:

$$\begin{array}{l|l} bird(X) \leftarrow chicken(X) & chicken(tina) \\ flies(X) \prec bird(X) & scared(tina) \\ \sim flies(X) \prec chicken(X) & \\ flies(X) \prec chicken(X), scared(X) & \end{array}$$

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 $\mathcal{A}_2 = \{ flies(tina) \prec bird(tina) \}$

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

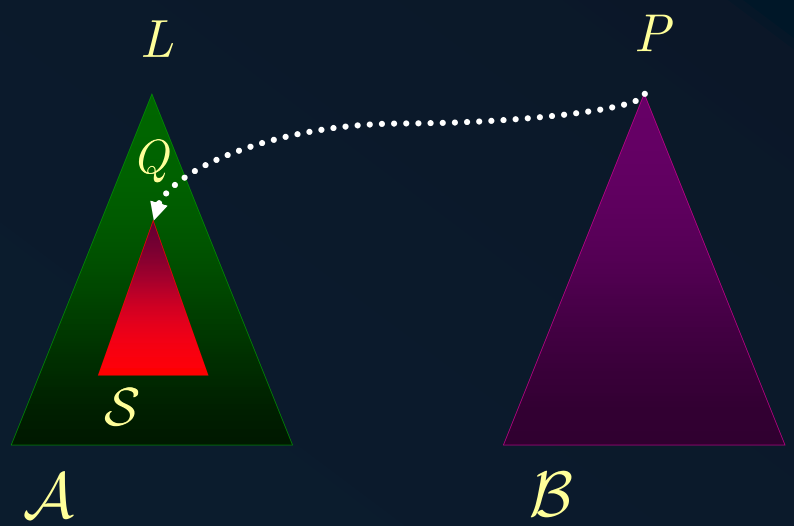
- $\mathcal{A}_3$ is preferred to $\mathcal{A}_1$ because it is more precise more information).
- $\mathcal{A}_1$ is preferred to $\mathcal{A}_2$ because it is more concise (direct).

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 $\langle \mathcal{S},\ Q \rangle$ of $\langle \mathcal{A},\ L \rangle$ 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:

$buy\_shares(X) \prec good\_price(X)$

$\sim buy\_shares(X) \prec risky(X)$

$risky(X) \prec in\_fusion(X, Y)$

$good\_price(acme)$

$in\_fusion(acme, estron)$

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

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

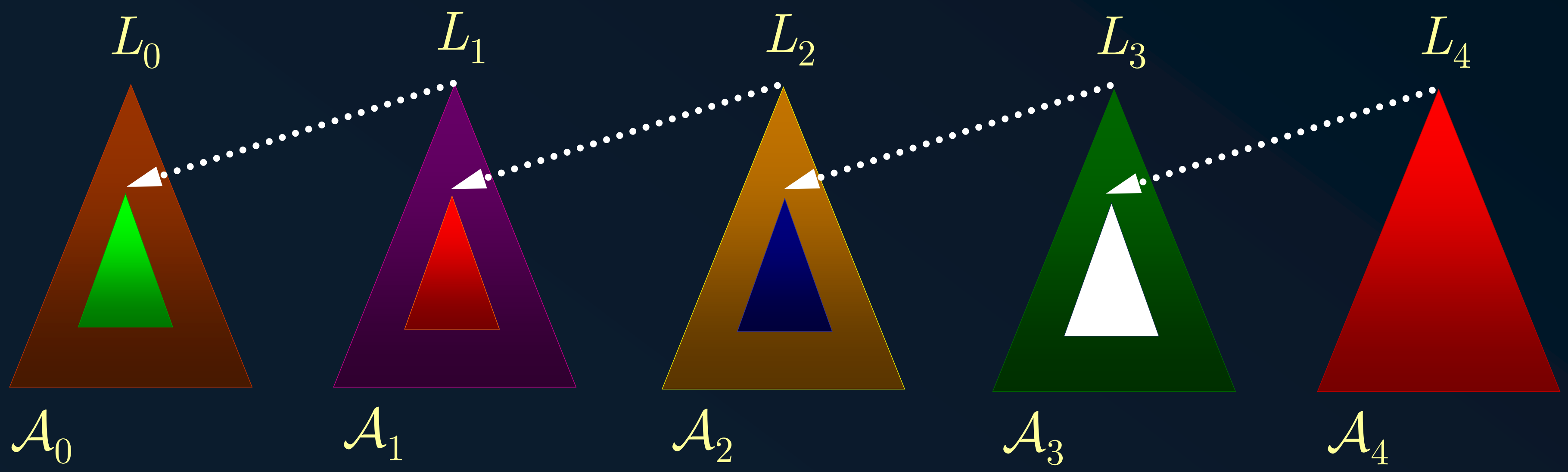
where $\mathcal{B} = \{ pacifist(nixon) \prec quaker(nixon) \}$

Conceptual View

Definition of Status of Arguments
Definition of Defeat among Arguments
Definition of Conflict among arguments
Definition of Argument
Definition of the Underlying (Logical) Language

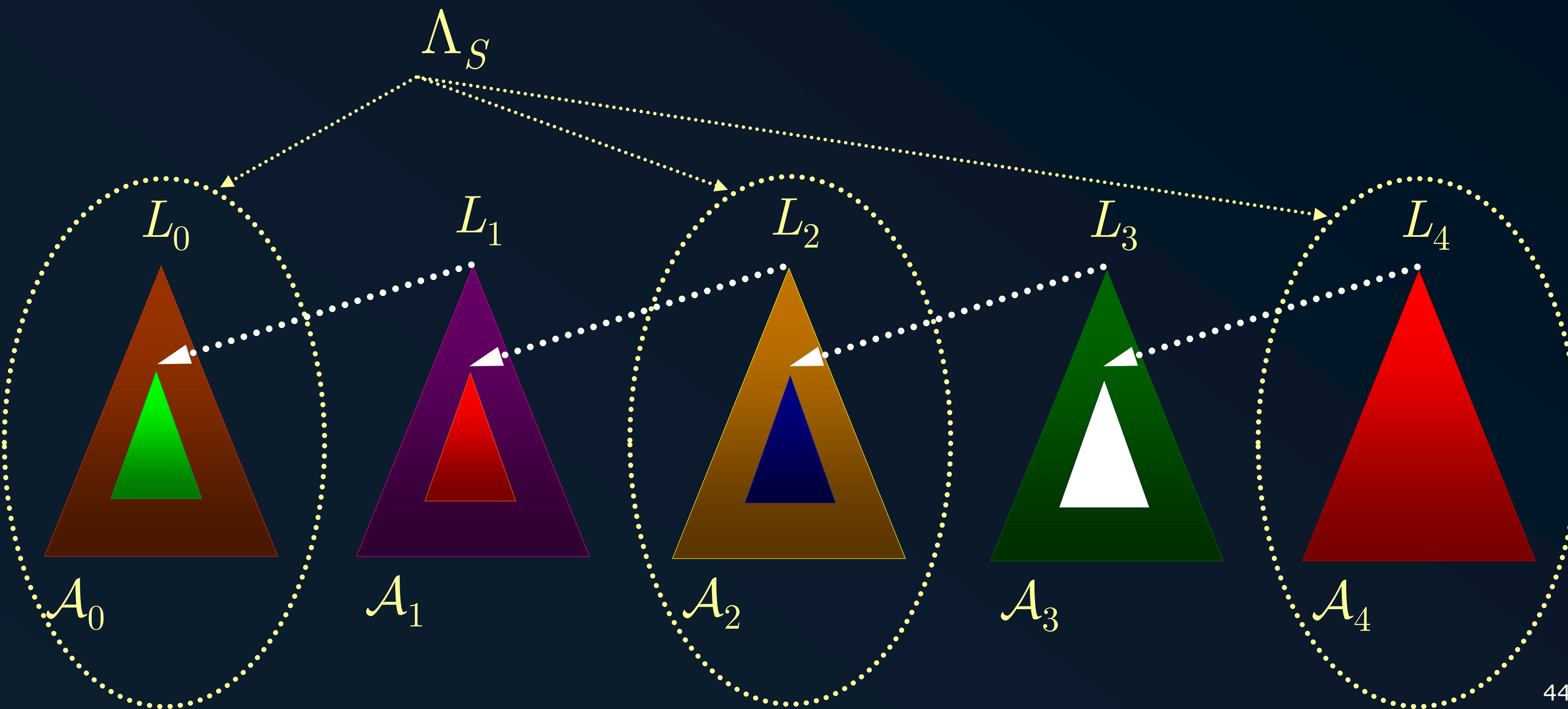
Argumentation Line

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 $\langle \mathcal{A}_{i\text{-}1}, L_{i\text{-}1} \rangle$.



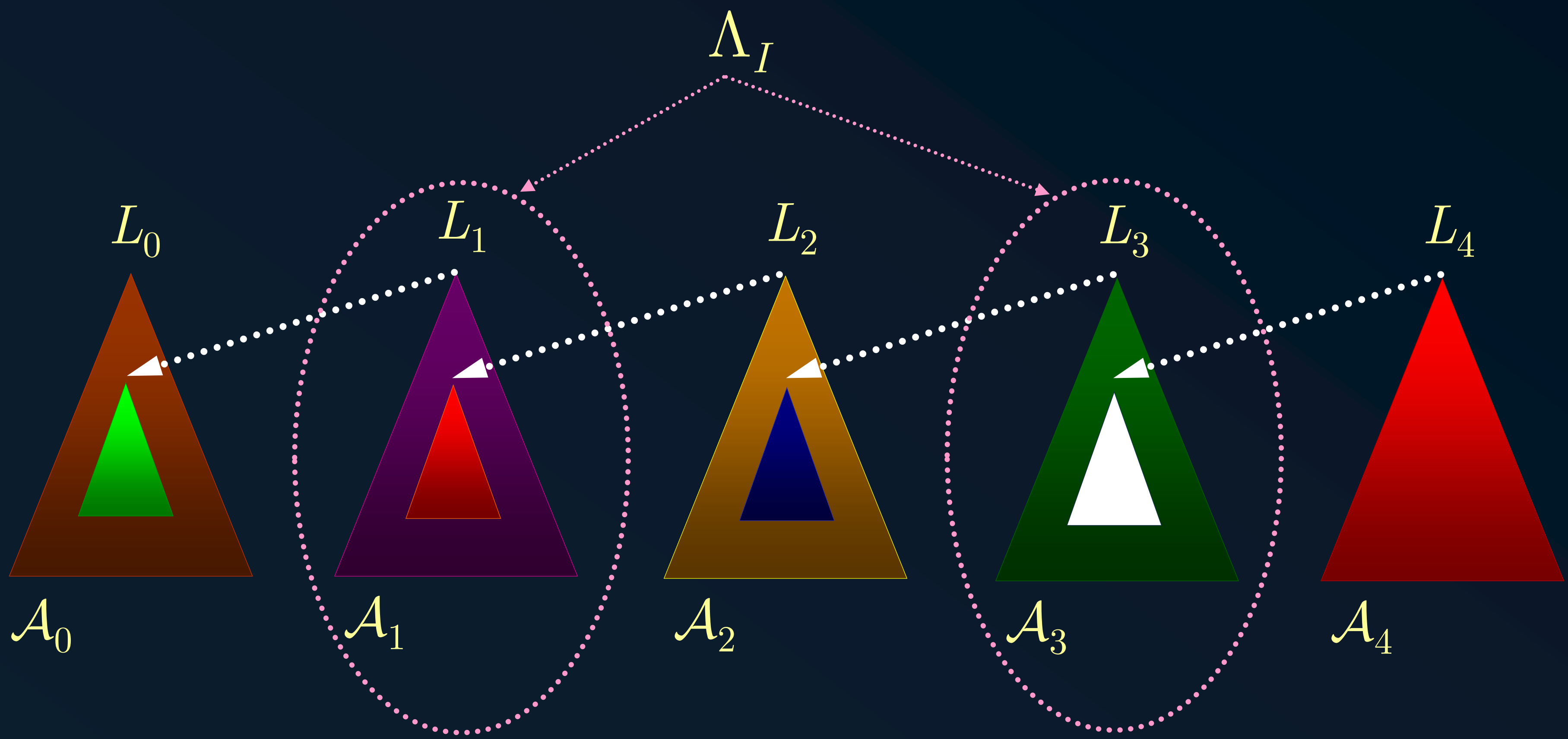
Argumentation Line

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 $\mathit{supporting}$ $\mathit{arguments}$ and $\Lambda_I = [\langle \mathcal{A}_1, L_1 \rangle, \langle \mathcal{A}_3, L_3 \rangle, \ldots]$ are $\mathit{interfering}$ $\mathit{arguments}$.



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 $\langle \mathcal{A}_1, L_1 \rangle$

$\langle \mathcal{A}_4, L_4 \rangle$ 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:

$$\Lambda_1 = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_1, L_1 \rangle, \langle \mathcal{A}_3, L_3 \rangle]$$

$$\Lambda_2 = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_2, L_2 \rangle, \langle \mathcal{A}_4, L_4 \rangle]$$

$$\Lambda_3 = [\langle \mathcal{A}_0, L_0 \rangle, \langle \mathcal{A}_2, L_2 \rangle, \langle \mathcal{A}_5, L_5 \rangle]$$

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:

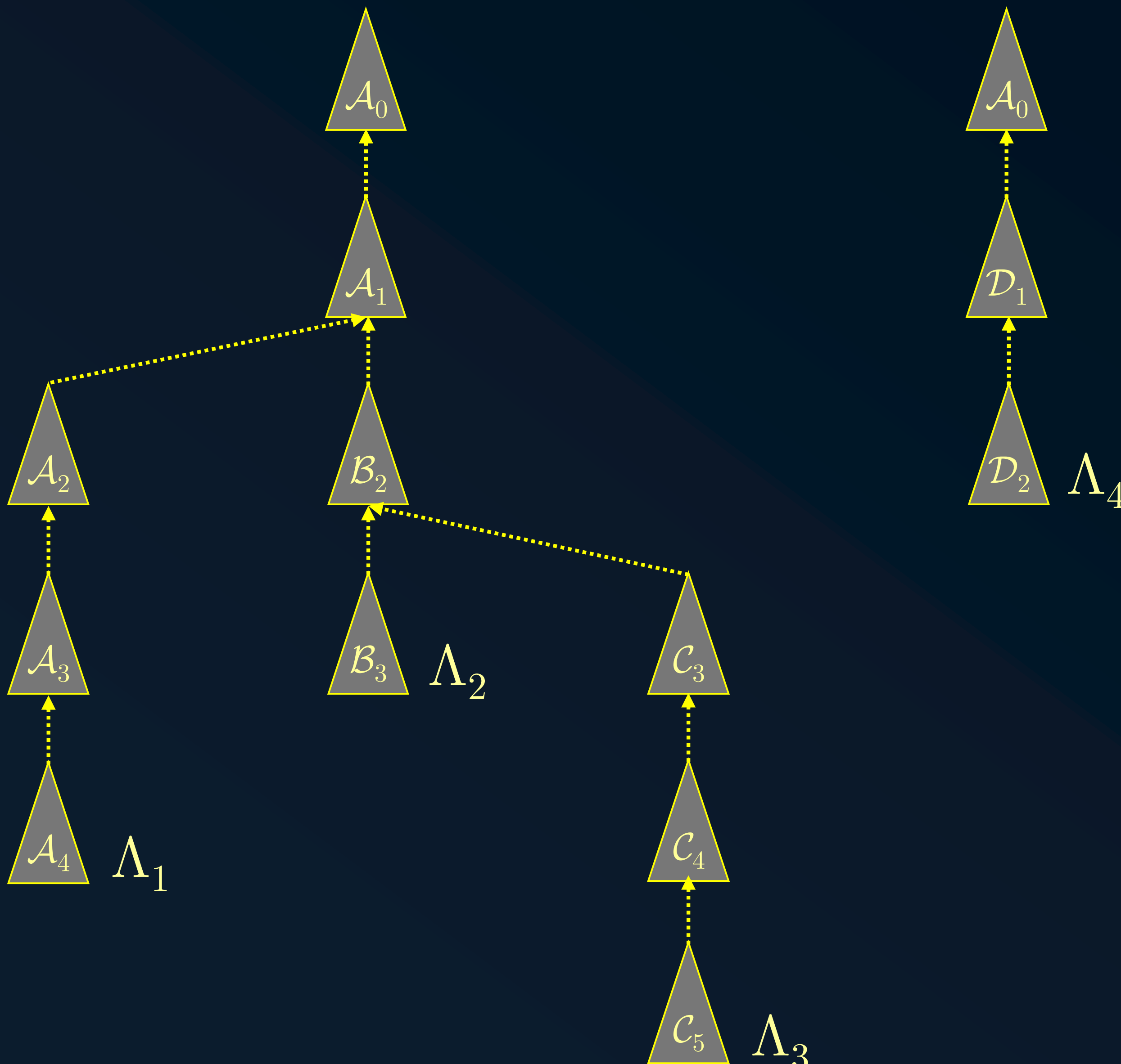
1. Λ is a finite sequence (no circularity).
2. The set Λ_S, of supporting arguments is concordant, and the set Λ_I, of interfering arguments is concordant.
3. There is no argument $\langle \mathcal{A}_k,\, L_k\rangle$ in Λ that is a subargument of a preceding argument $\langle \mathcal{A}_i,\, L_i\rangle$, $i < k$.
4. For all i, such that $\langle \mathcal{A}_i,\, L_i\rangle$ is a blocking defeater for $\langle \mathcal{A}_{i\text{-}1},\, L_{i\text{-}1}\rangle$, 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

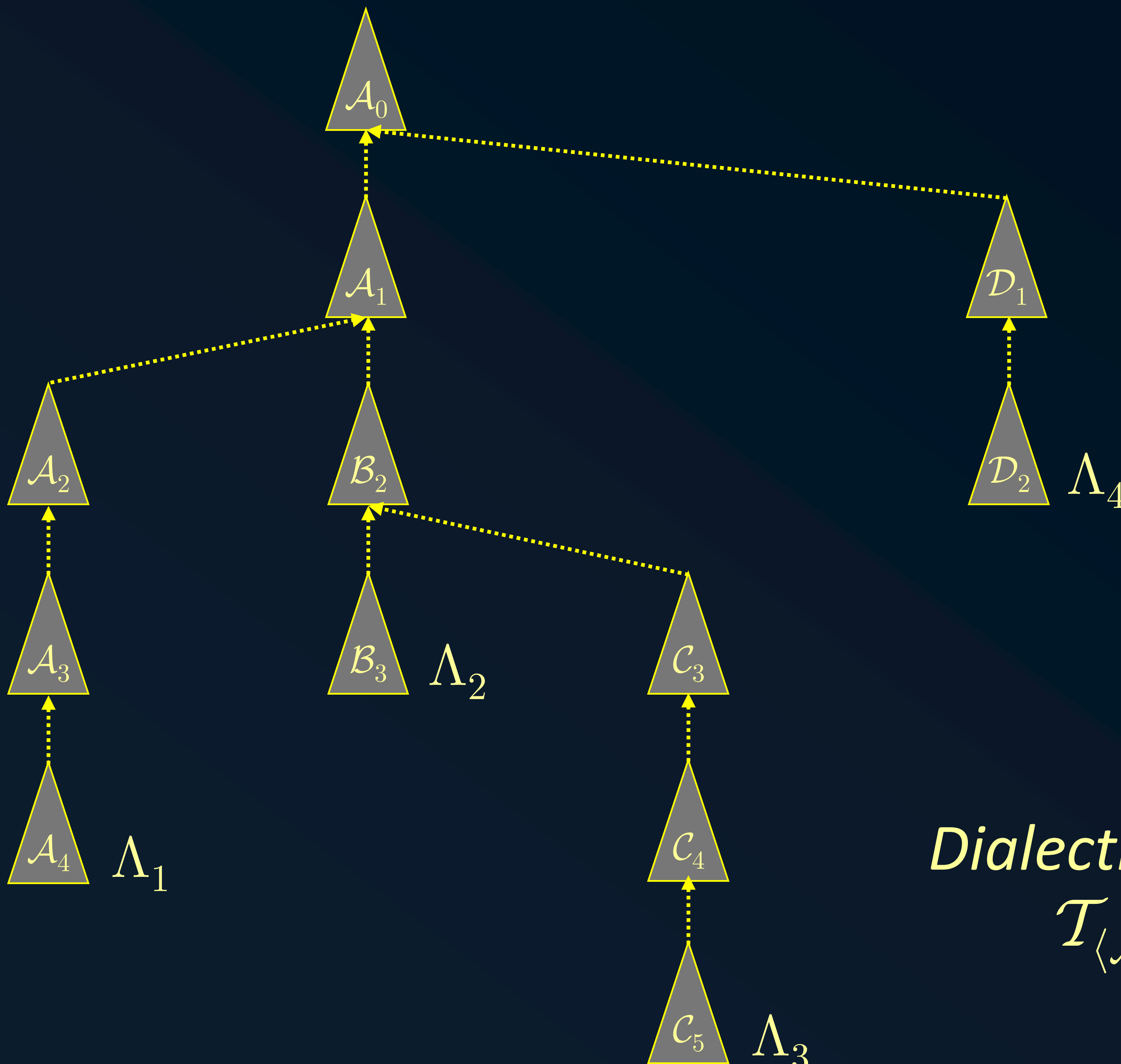
Argumentation Lines (Bundle)

Argumentation Lines (Bundle)

Argumentation Lines (Bundle)



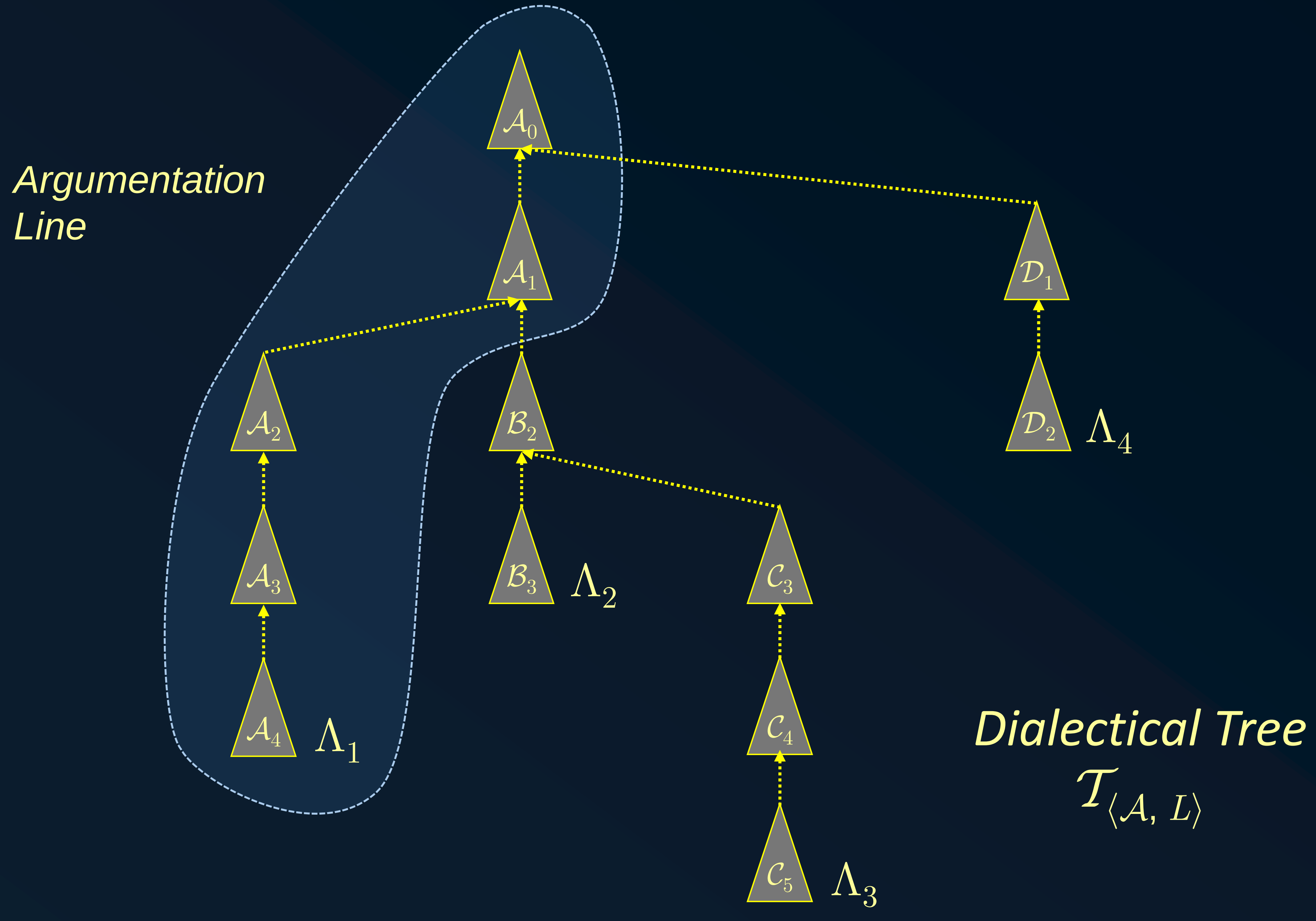
Argumentation Lines (Bundle)



Dialectical Tree

$\mathcal{T}_{\langle \mathcal{A},\, L \rangle}$

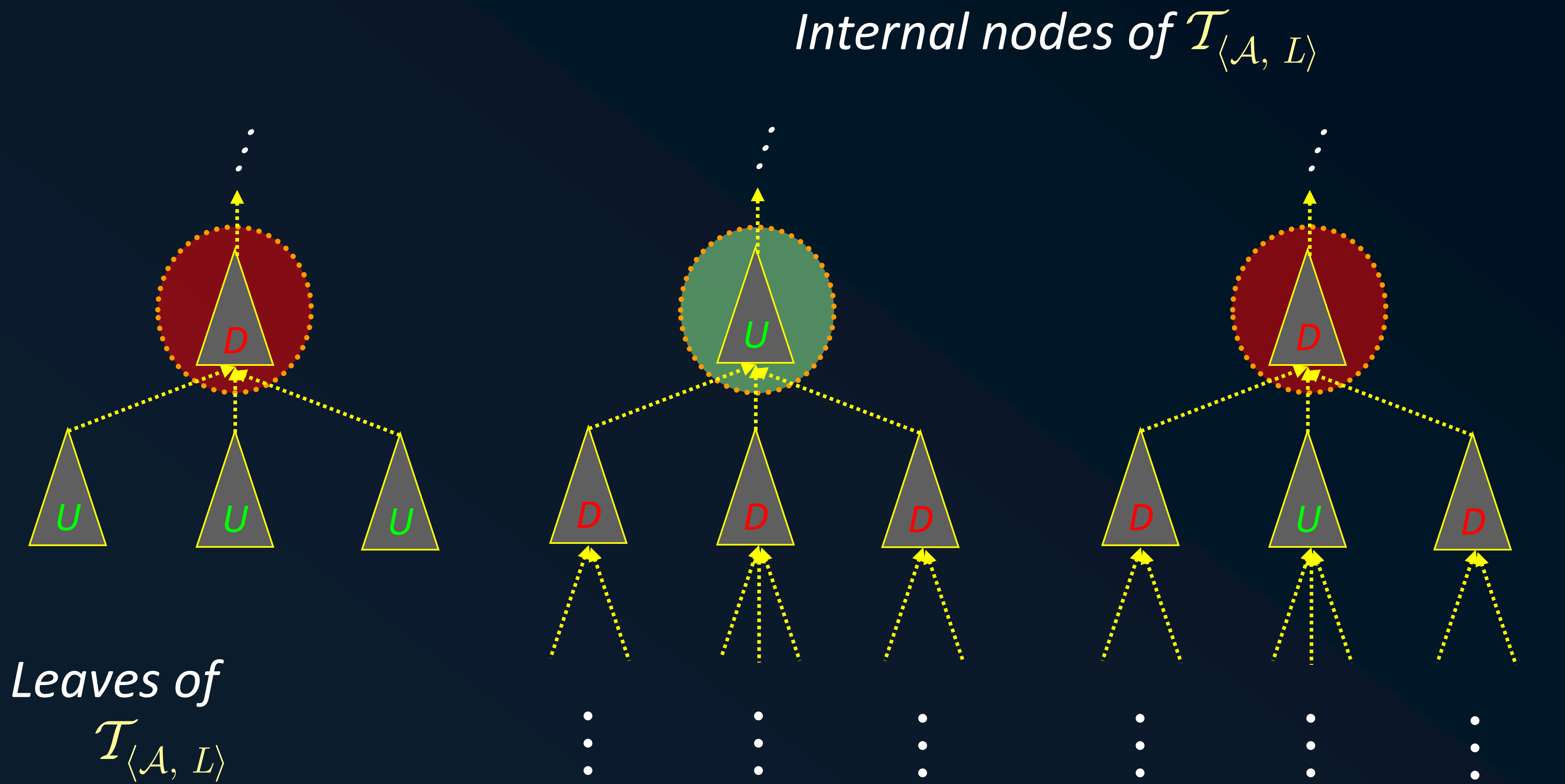
Argumentation Lines (Bundle)



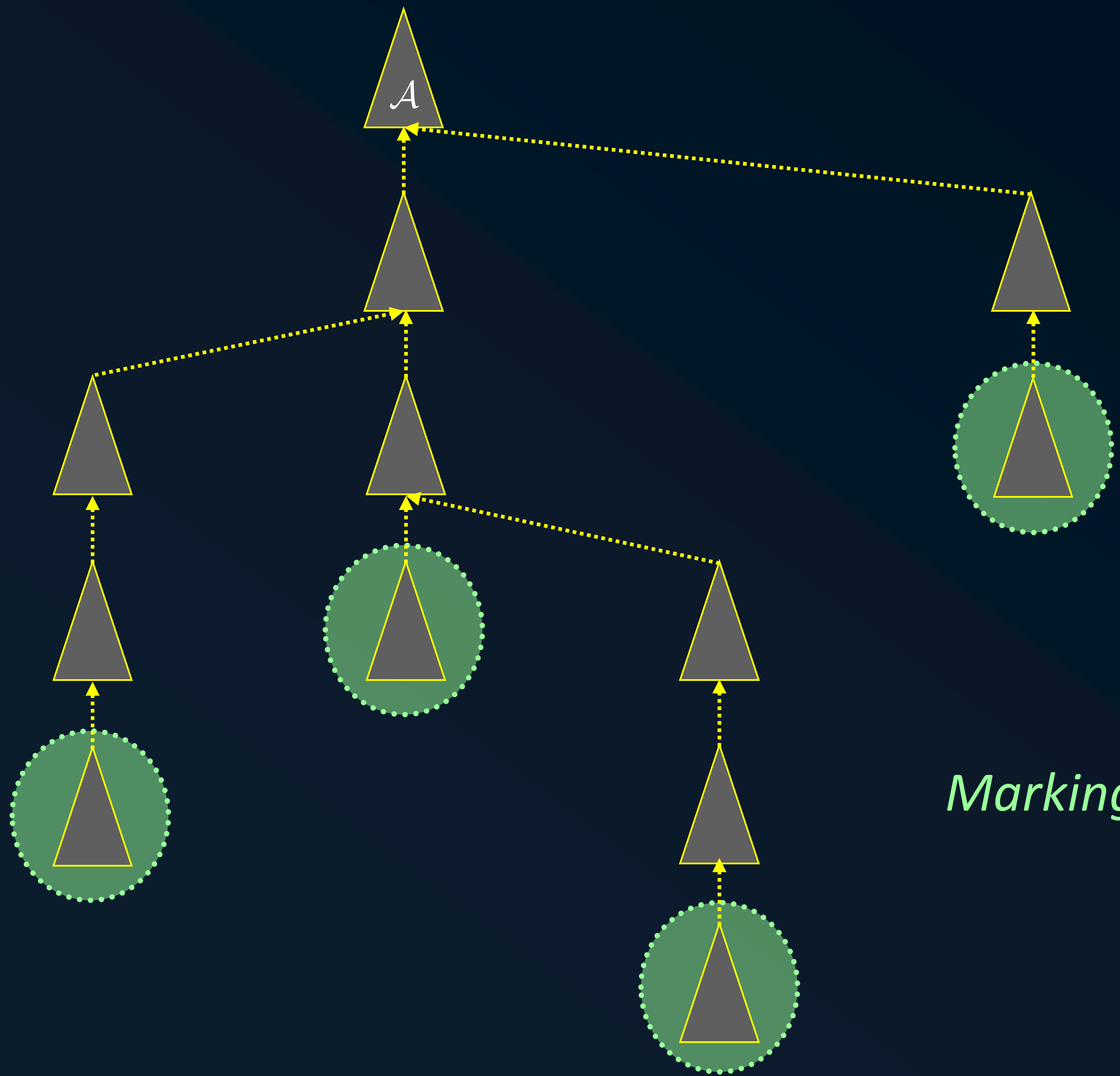
Dialectical Tree

- 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 admissible, and therefore finite, every dialectical tree is also finite.

Marking of a Dialectical Tree

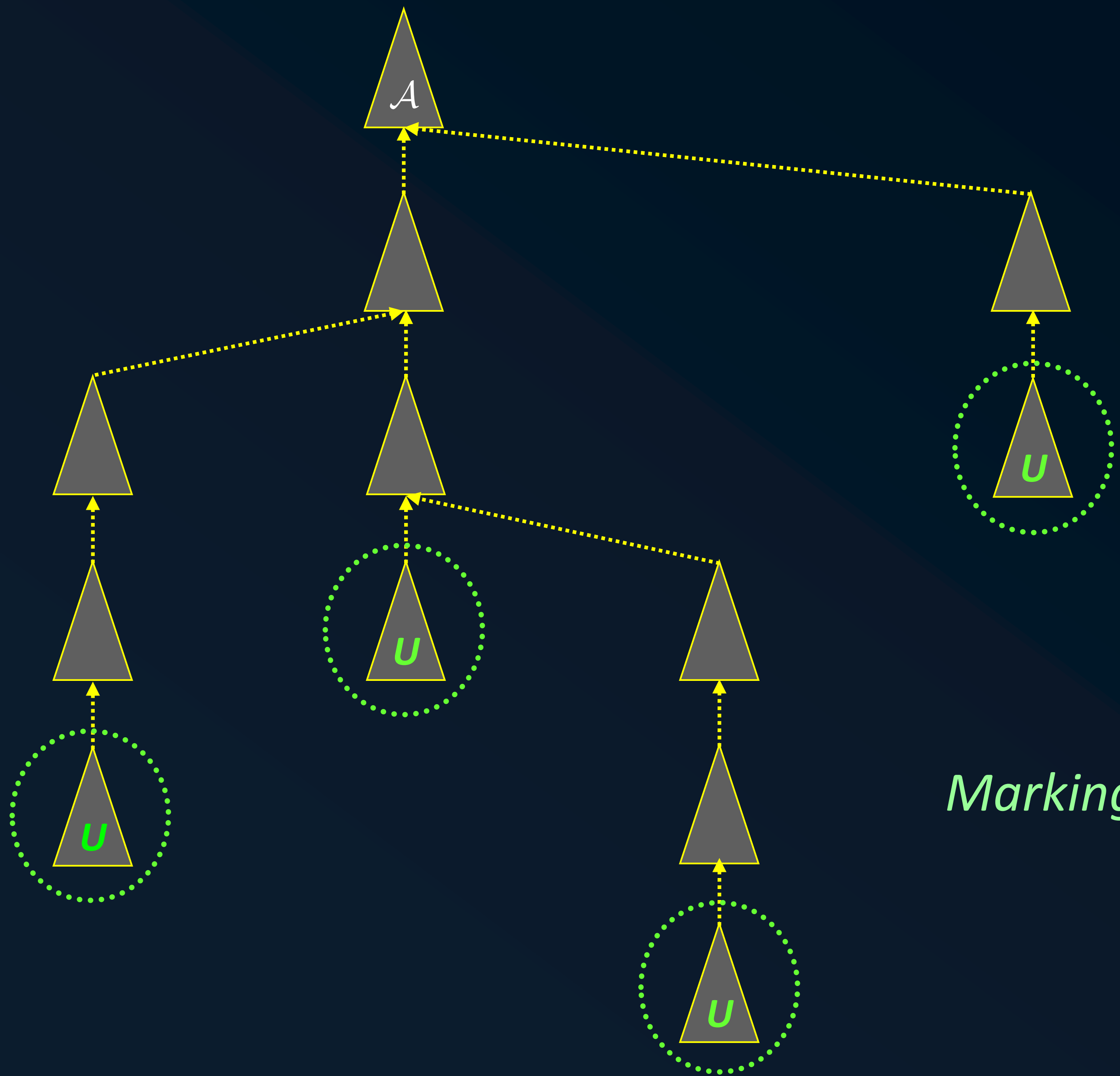


Dialectical Tree



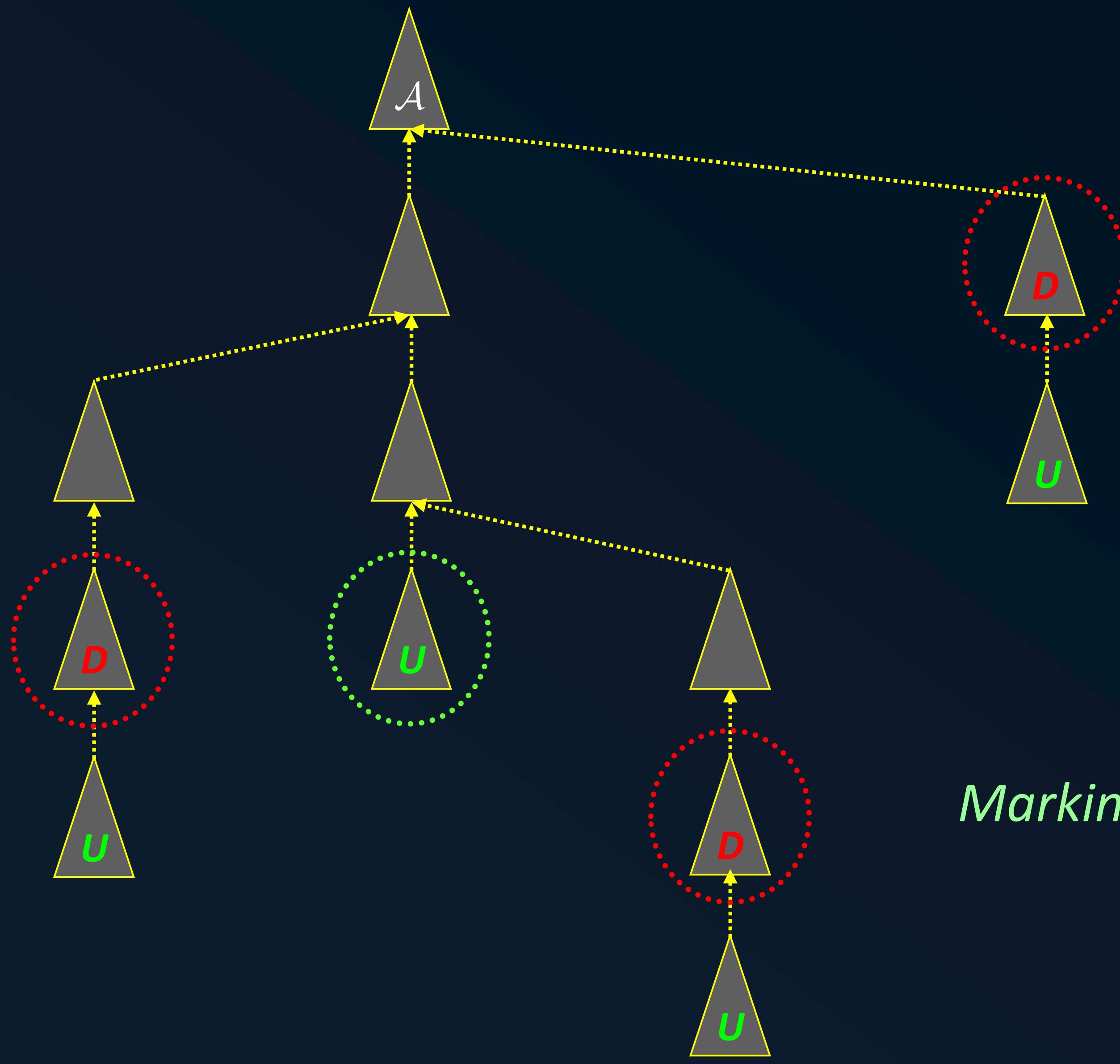
Marking

Dialectical Tree



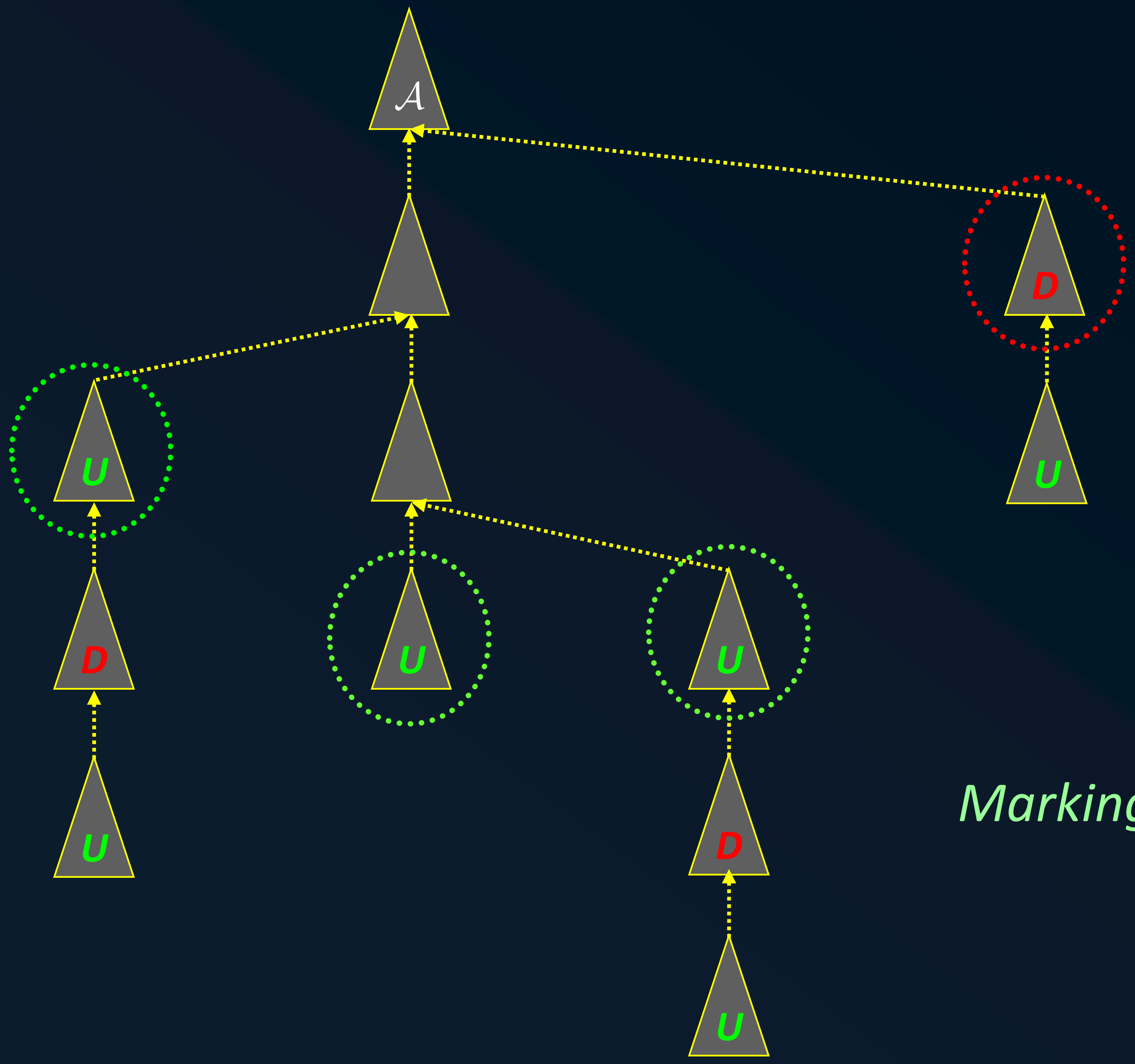
Marking

Dialectical Tree



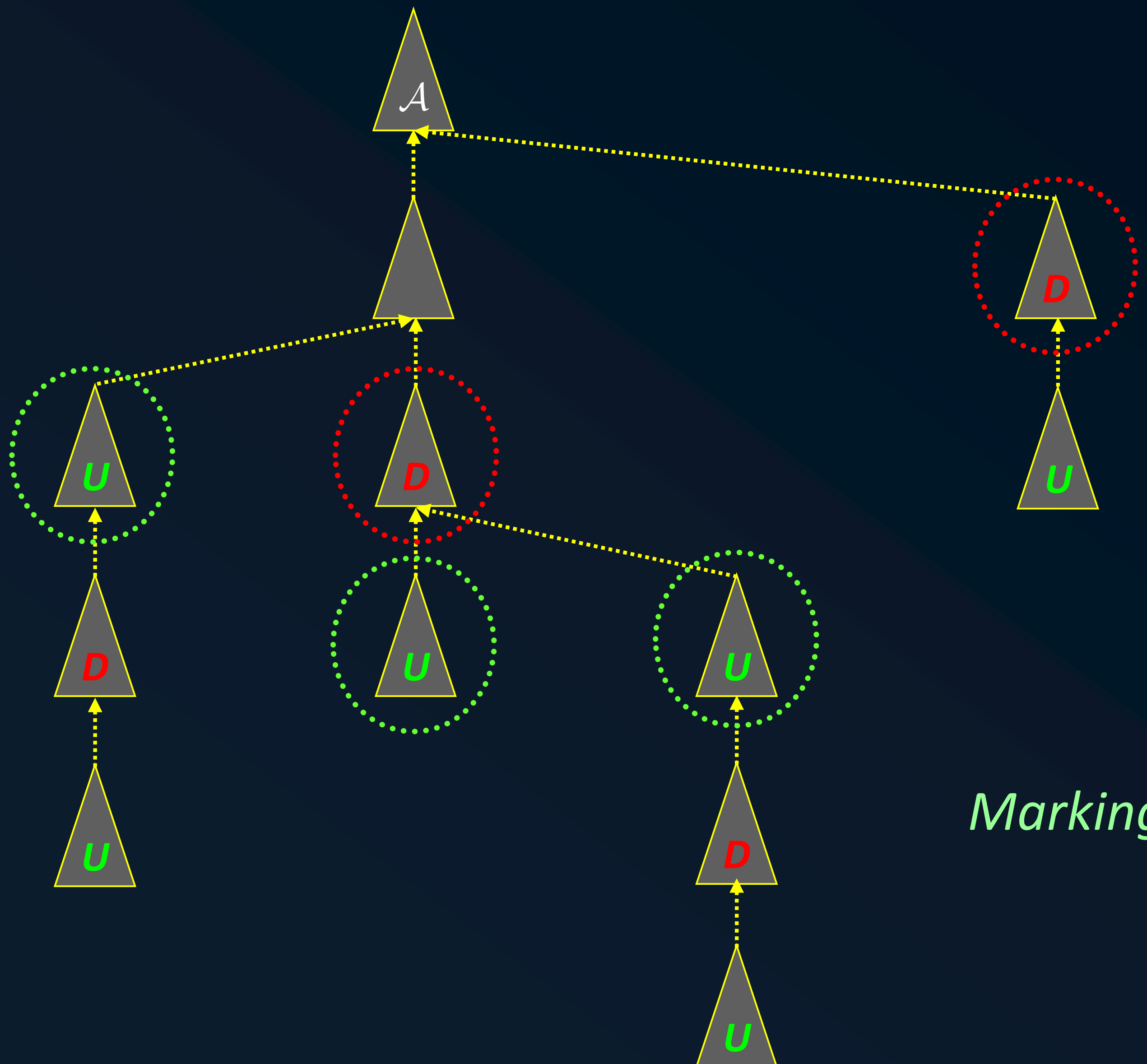
Marking

Dialectical Tree



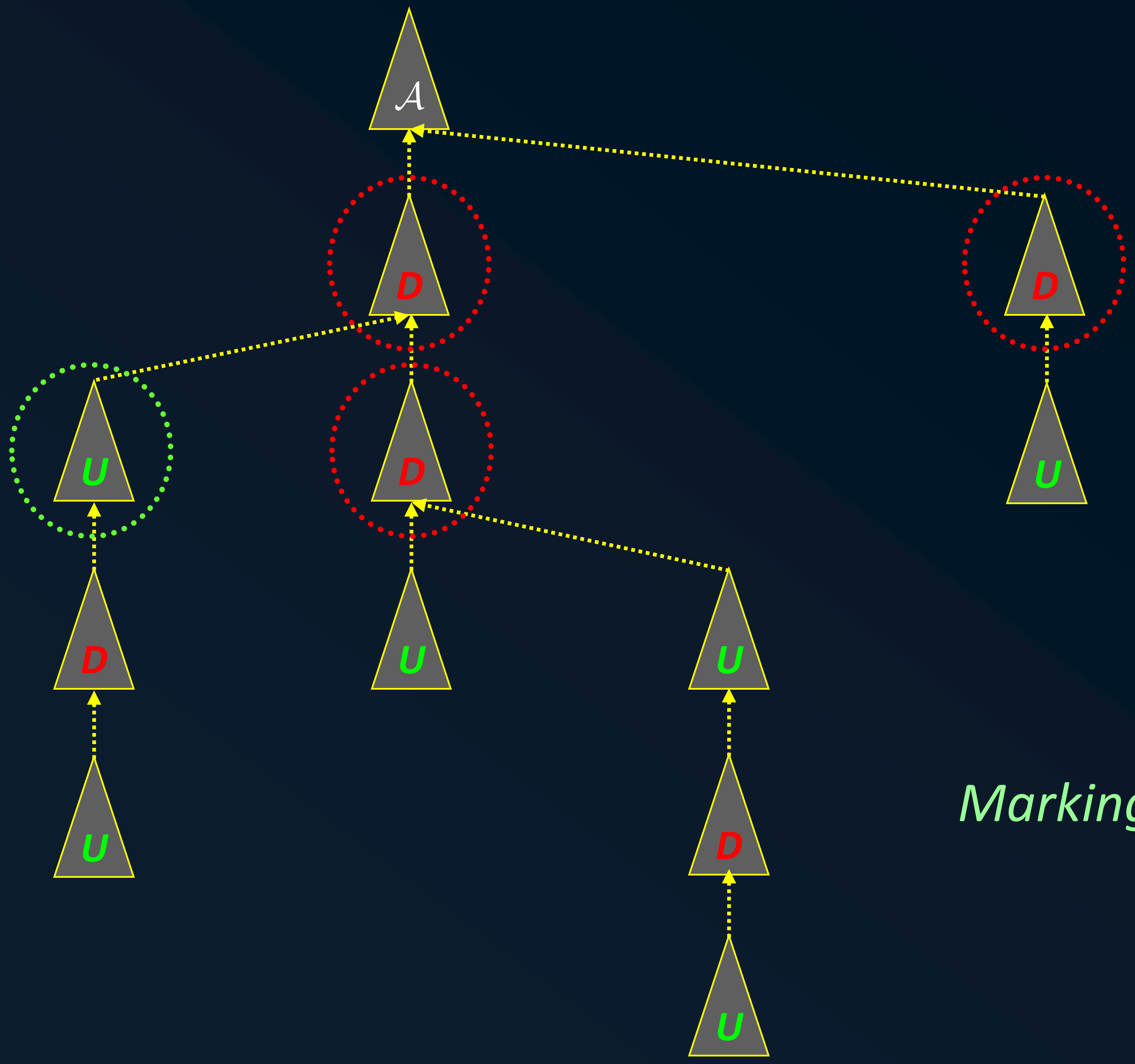
Marking

Dialectical Tree



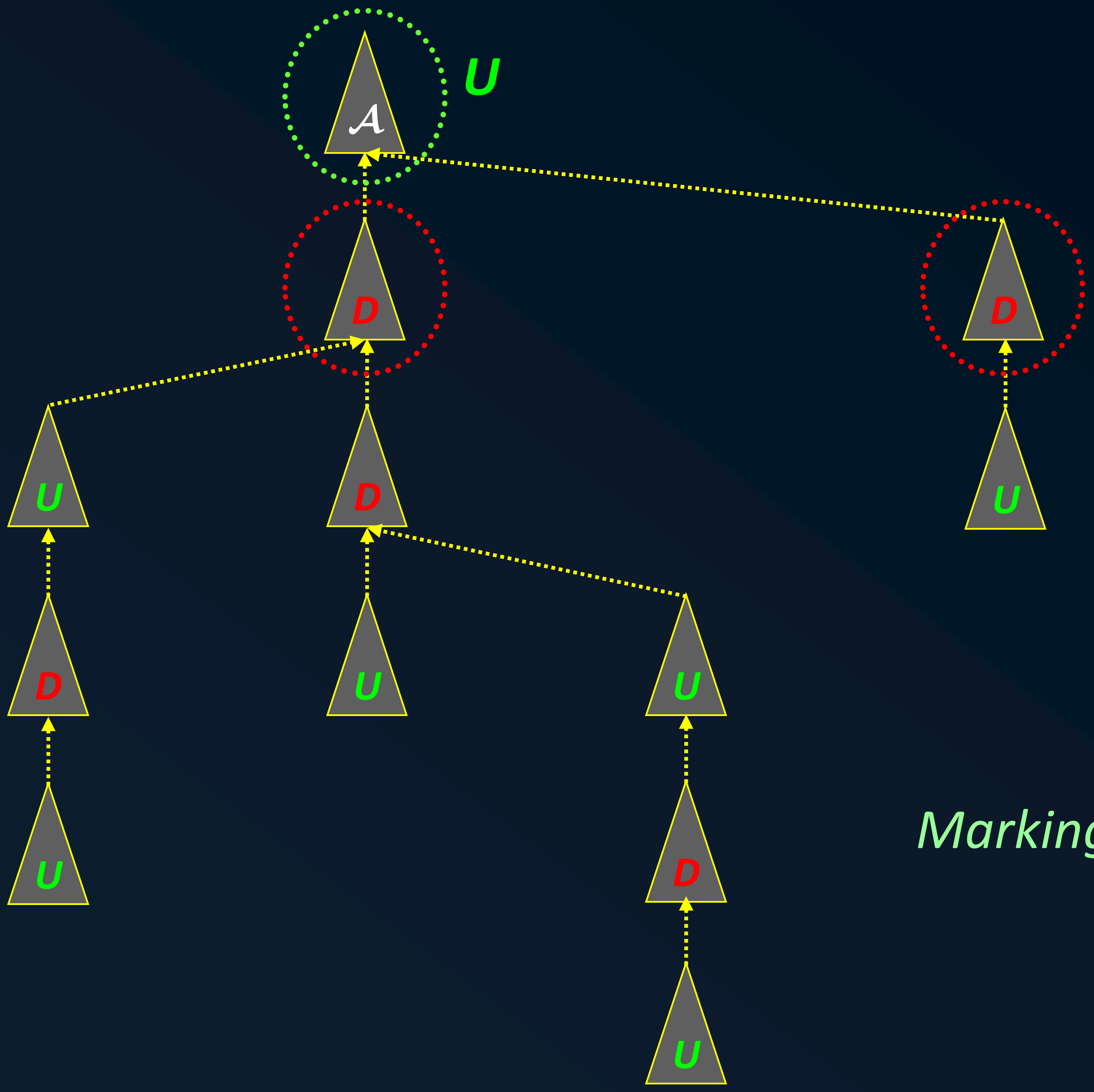
Marking

Dialectical Tree



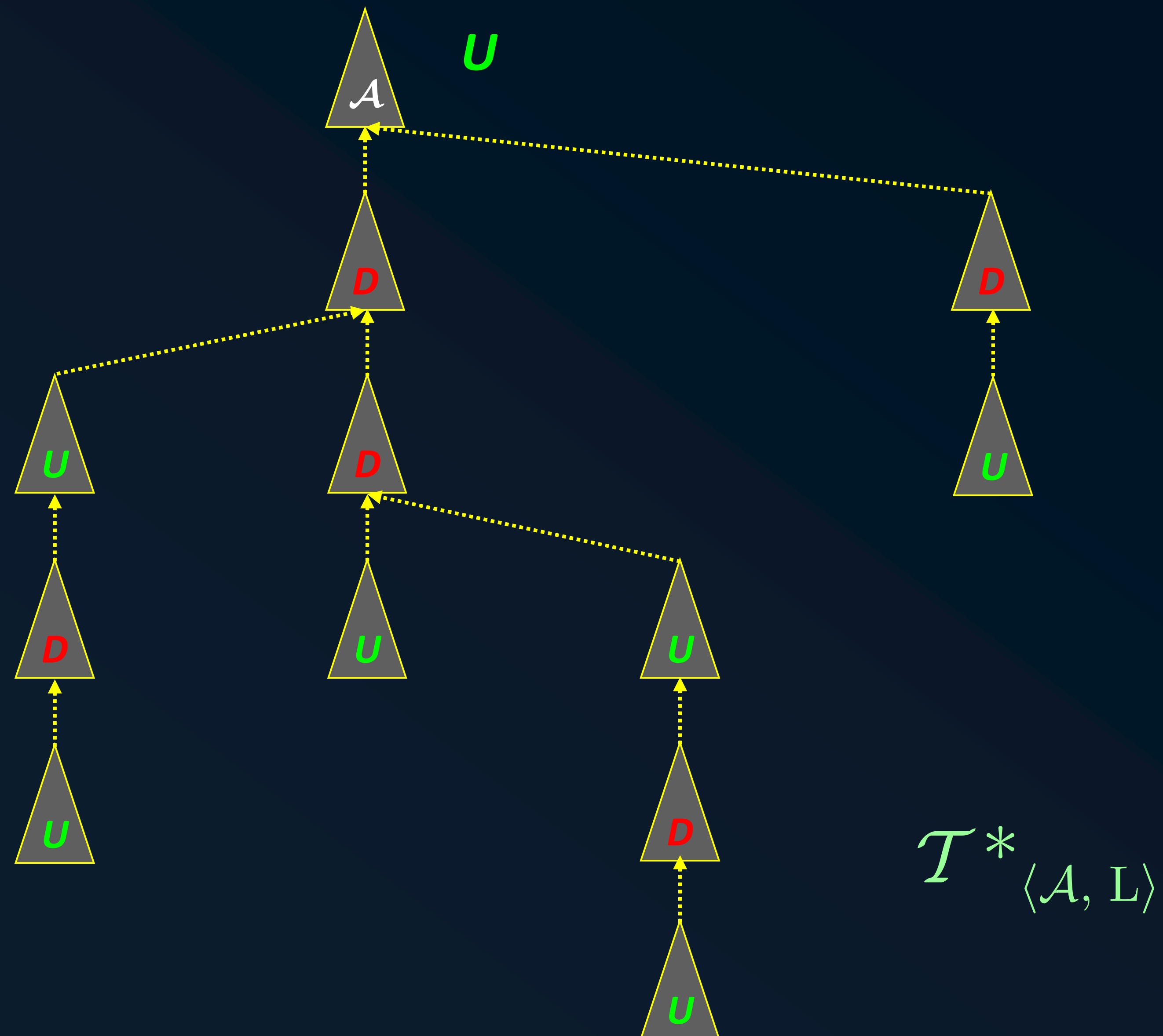
Marking

Dialectical Tree



Marking

Dialectical Tree (Marked)



Warranted Literals

- Let $\mathcal{P} = (\Pi, \Delta)$ be a defeasible program.

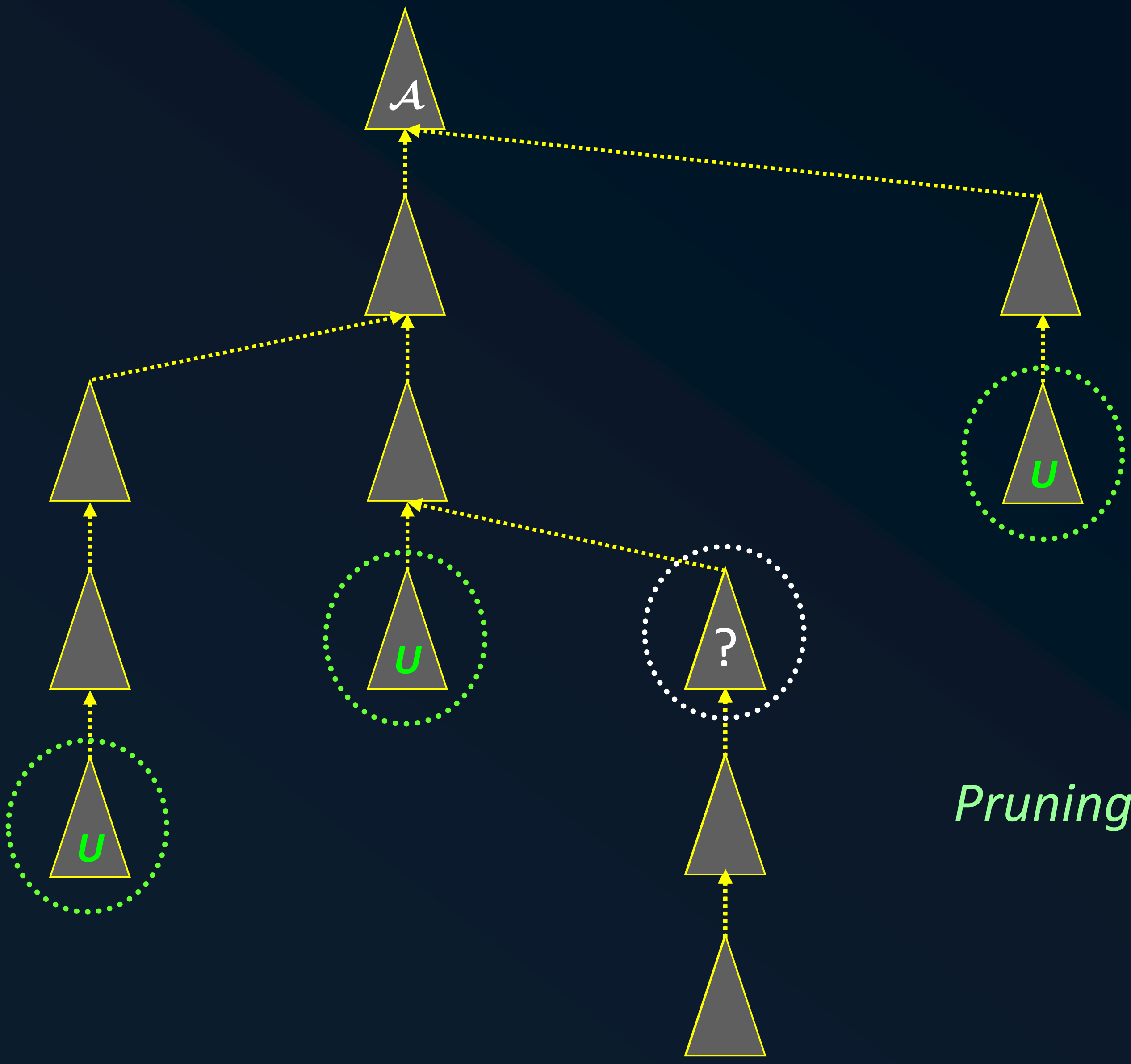
  Let $\langle \mathcal{A}, L \rangle$ be an argument and let $\mathcal{T}*_{\langle \mathcal{A}, L \rangle}$ be its associated dialectical tree.

  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.

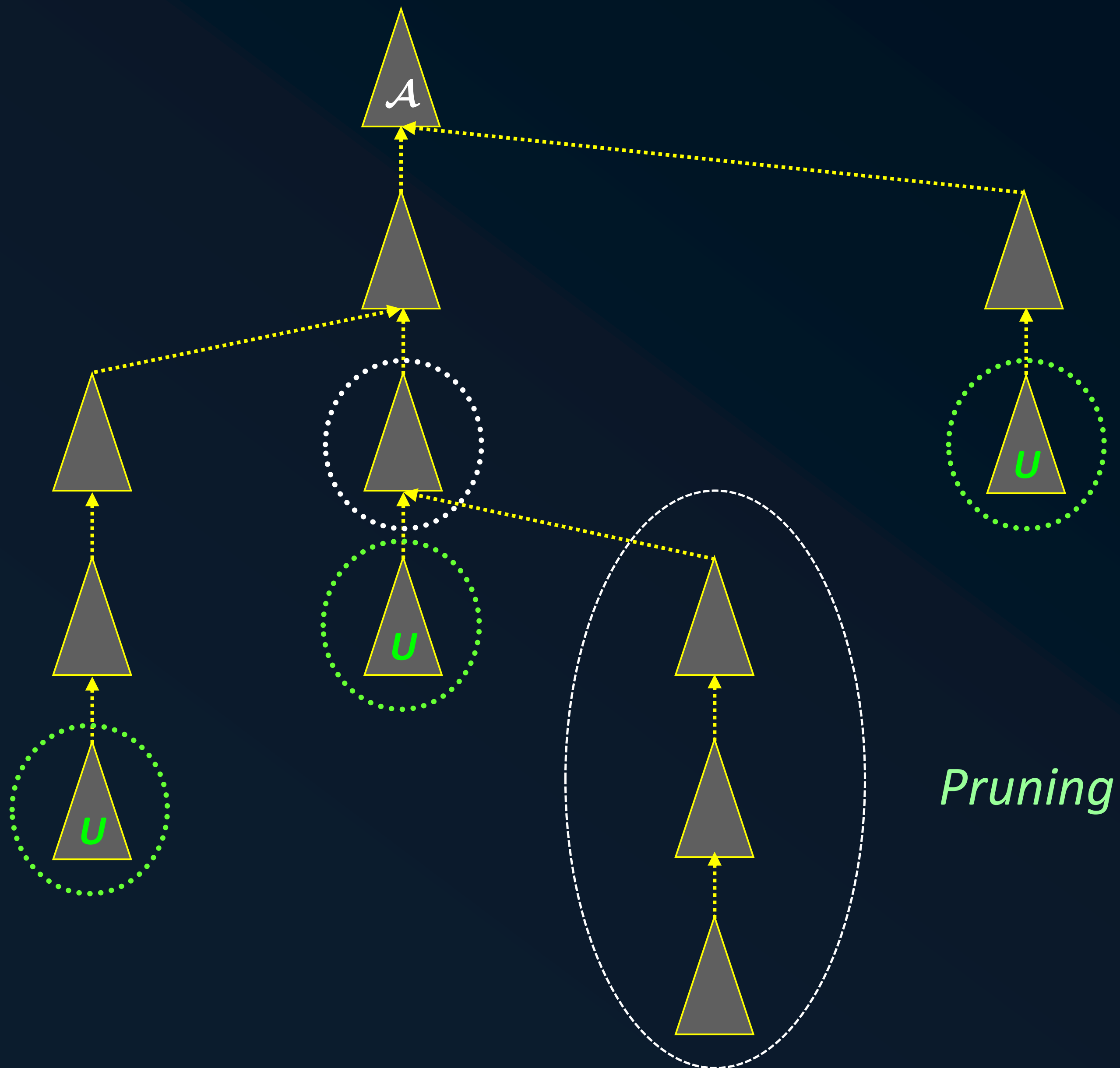
- We will say that $\mathcal{A}$ *is a warrant for* L.

Dialectical Tree: Pruning

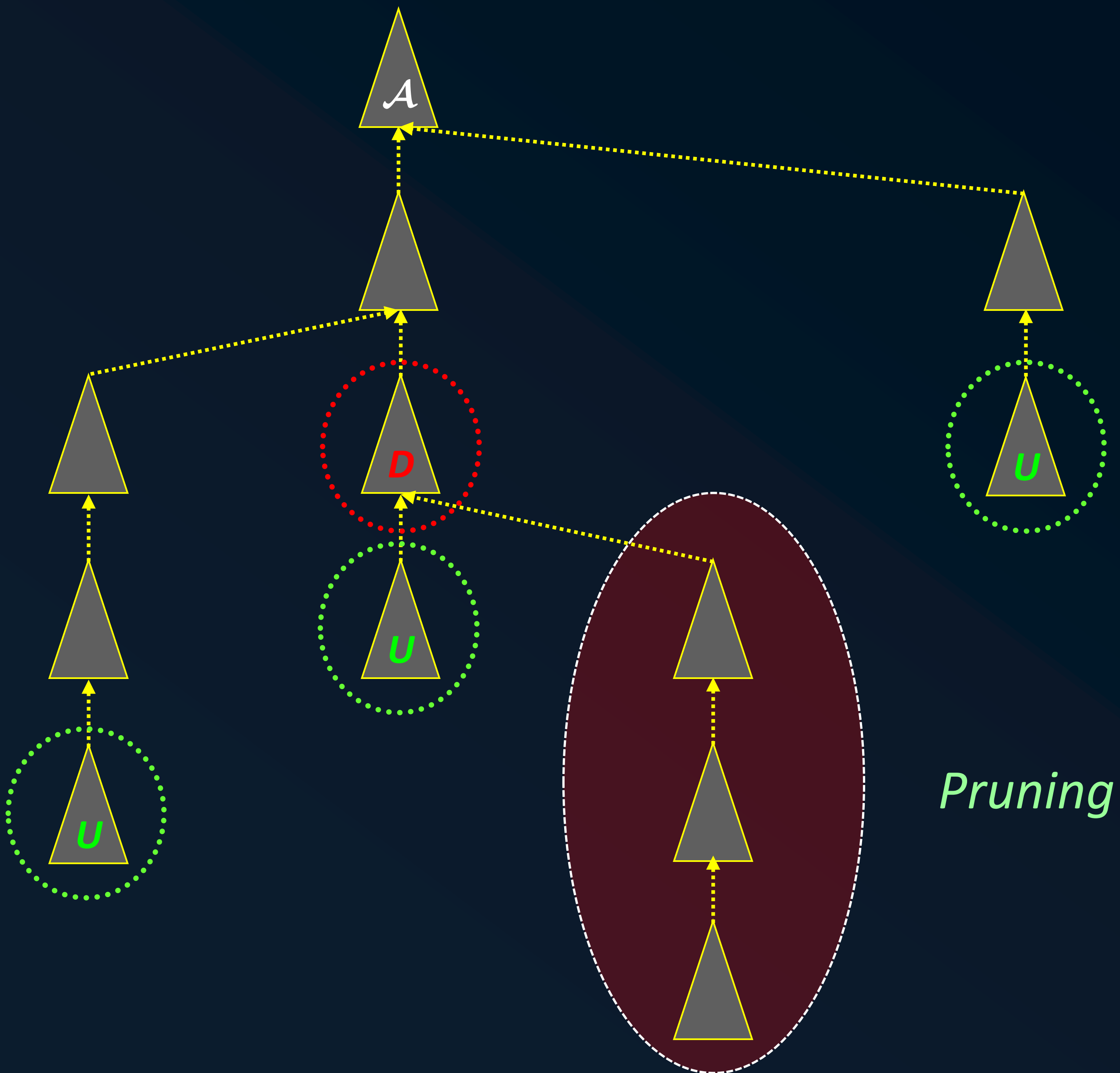


Pruning

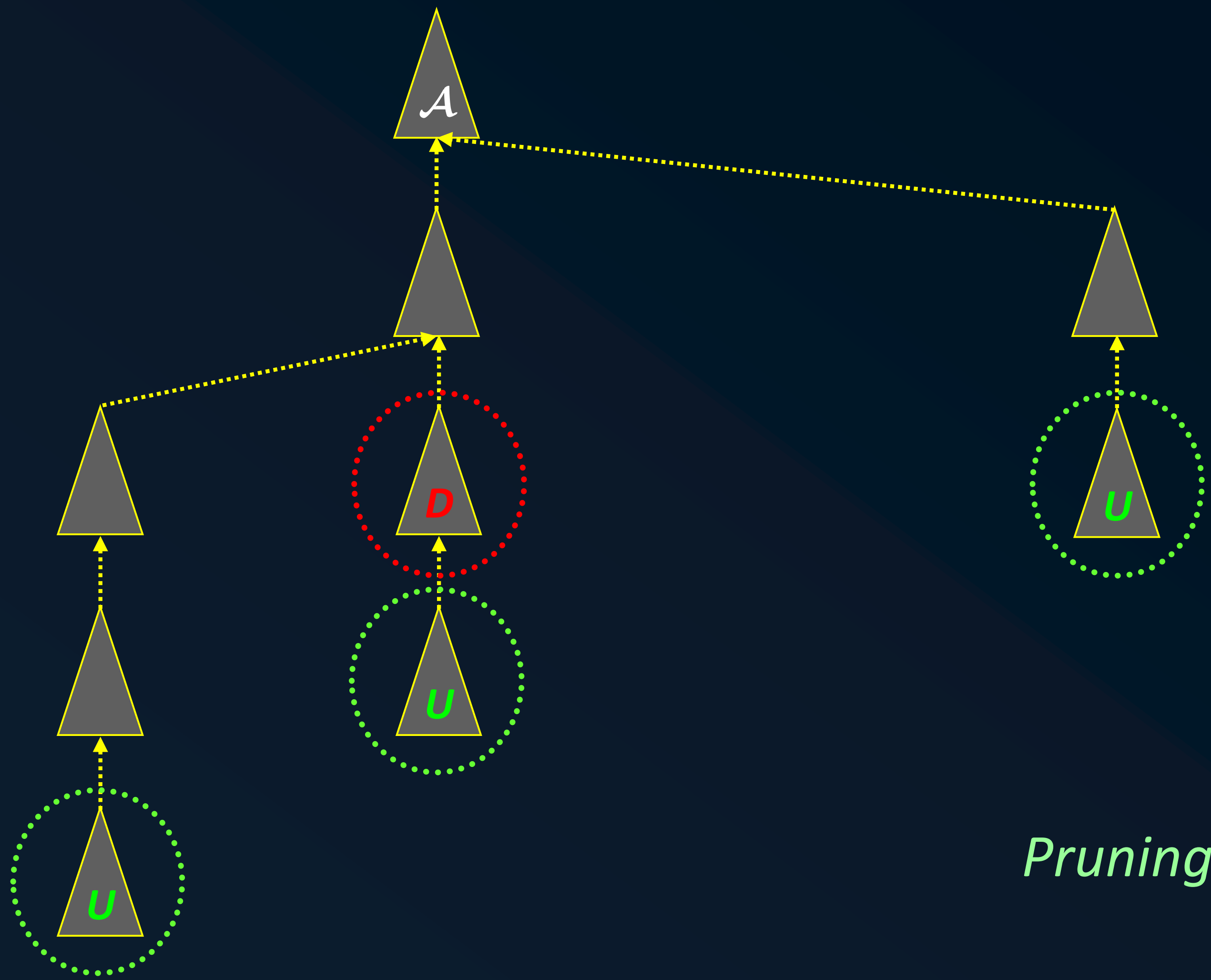
Dialectical Tree: Pruning



Dialectical Tree: Pruning



Dialectical Tree: Pruning



Pruning

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
```

https://shorturl.at/iqKQX

https://hosting.cs.uns.edu.ar/~daqap/client/index.html

Further topics

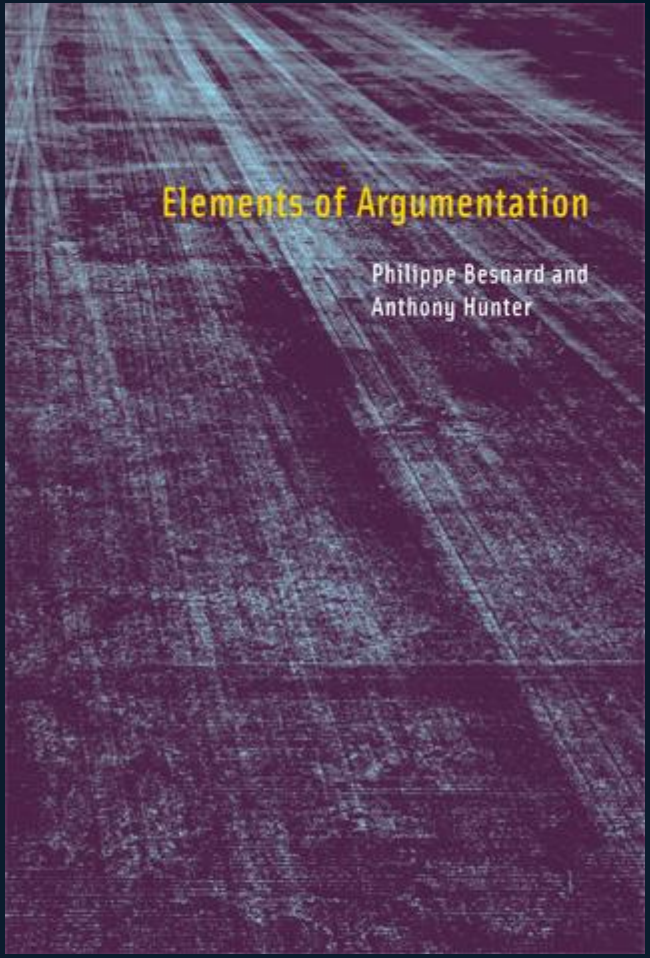
- *Argumentation and decision making (e.g., Amgoud et al, Atkinson, Bench-Capon).*
- *Argumentation and negotiation (e.g., Parsons, McBurney, Rahwan).*
- *Argumentation with uncertainty formalisms (e.g., Simari, Chesñevar, Godo).*
- *Dialogical argumentation in multiple agents (e.g., Prakken, Parsons, Amgoud, Wooldridge, McBurney, Rahwan, Toni, Sadri, Torroni, Maudet, Kakas, Moratis, Black and Hunter).*
- *Implementations (e.g., Dungine and DLV systems for abstract argumentation, ASPIC, DeLP and ABA systems for defeasible logics, and connection-graph systems for classical logic).*
- *Applications (e.g., law, medicine, e-commerce, etc.).*

References

- J. Pollock. Defeasible Reasoning, Cognitive Science, 11, 481-518, 1987.
- G. R. Simari, R. P. Loui. A Mathematical Treatment of Defeasible Reasoning and Its Implementation, Artificial Intelligence, 53, 125-157, 1992.
- P. Dung. *On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games.* Artificial Intelligence, 77(2):321-358, 1995.
- G. Vreeswijk, Abstract Argumentation Systems, Artificial Intelligence, 90, 225-279, 1997.
- C. Chesñevar, A. Maguitman, R. Loui. "*Logical Models of Argument*. ACM Computing Surveys, 32(4):337-383, 2000.
- H. Prakken, G. Vreeswijk. Logical Systems for Defeasible Argumentation, in D. Gabbay (Ed.), Handbook of Philosophical Logic, 2nd Edition, 2002.
- A. J. García, G.R. Simari. *Defeasible Logic Programming: An Argumentative Approach,* Theory and Practice of Logic Programming. Vol 4(1), 95-138, 2004.
- C. Chesñevar; G. Simari; T. Alsinet; L. Godo - *A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge* . Procs. of UAI-2004, Canada, 2004.
- G. R. Simari, A. García, M. Capobianco. Actions, Planning and Defeasible Reasoning. In Proc. 10th Intl. NMR 2004, Whistler BC, Canada. Pp. 377-384, 2004.
- M. Capobianco, C. Chesñevar, G. R. Simari. *Argumentation and the Dynamics of Warranted Beliefs in Changing Environments.* In Intl. Journal on Autonomous Agents and Multiagent Systems, 2005.
- I. Rahwan, G. R. Simari, *Argumentation in Artificial Intlligence*, 2009, Springer.
- P. Besnard, A. Hunter, *Elements of Argumentation*, 2008, MIT Press.

References

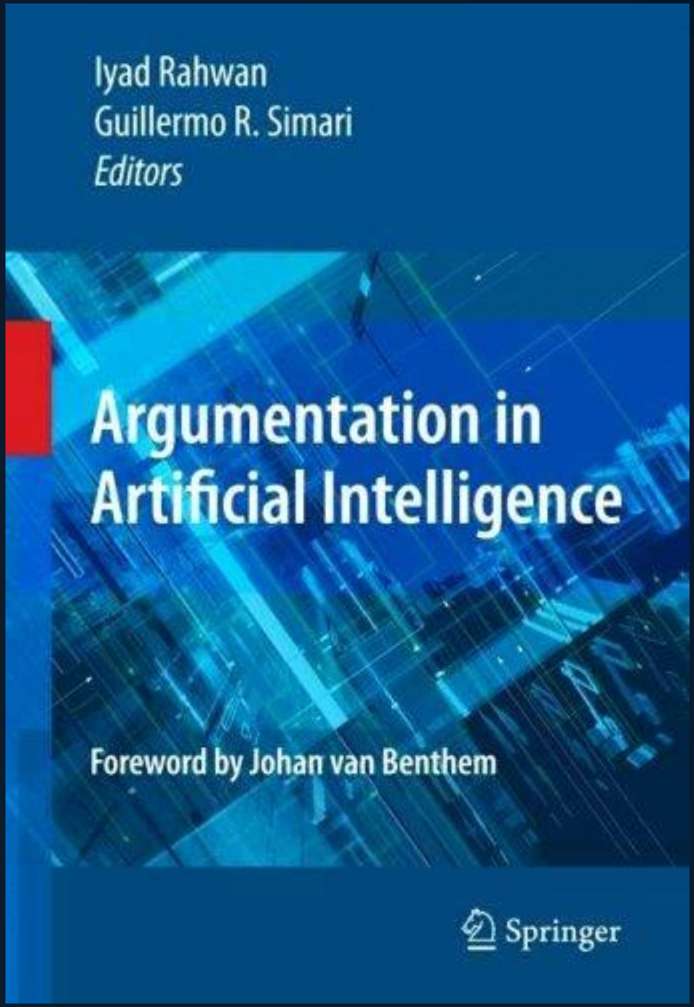
Chapter 2, *Abstract Argumentation*, of the book:



Elements of Argumentation
Philippe Besnard and Anthony Hunter
MIT Press, 2008
ISBN: 978-0-262-02643-7

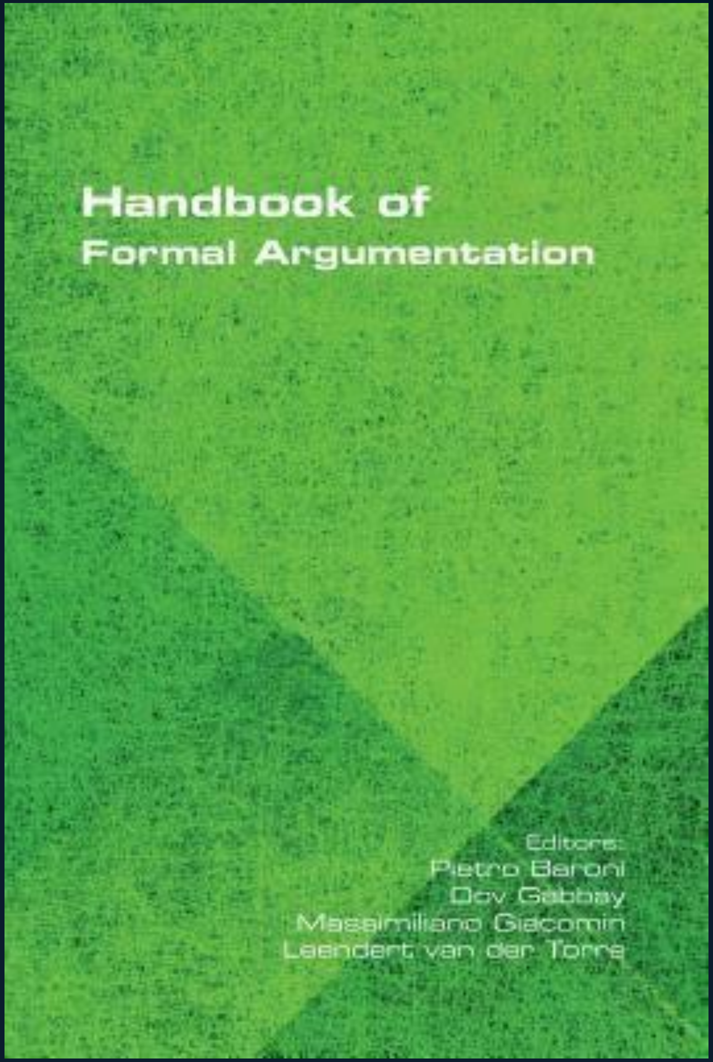
References

Several Chapters of the book:



Argumentation in Artificial Intelligence
Iyad Rahwan and Guillermo R. Simari
Springer, 2009
ISBN: 978-0-387-98196-3

References



Handbook of Formal Argumentation
Pietro Baroni, Dov Gabbay, Massimilino Giacomin
College Publications, Feb 28, 2018.

Thank you!