ABA

Assumption-Based Argumentation

ABA DEFINITION

ABA is a computational framework designed to generalize and bring together most existing approaches to default reasoning.

1. Inspired by how negation-as-failure works in logic programming.

2. Connects to Dung's abstract argumentation (AA), where arguments and attacks are abstract, and AA's semantic notions of "acceptability" apply.

DIFFERENCES TO AA

In AA, arguments and attacks are primitive

In ABA, arguments are deductions using inference rules, and these deductions are supported by assumptions.

ARGUMENTS

- Arguments are **deductions** of a claim that are supported by a set of assumptions.
- **Inference rules** are used to make deductions.
- Assumptions are sentences that are **open to challenge**.

Examples of Assumptions

Uncertain beliefs ("it will rain")

Unsupported beliefs ("I believe X")

Decisions ("perform action A")

Rule: If Source E is an expert about A, and E asserts A is true, and we assume "arguably(A)", then A may plausibly be taken as true.

The arguably(A) part is the defeasible condition, treated as an assumption.

ARGUMENTS FORMAL DEFINITION

- An argument for a claim c supported by a set of assumptions S is a tree
- The root of the tree is the claim c.
- Leaves are either assumptions from S or a special symbol τ
- Internal nodes are connected by inference rules: a node (conclusion) is the parent, and its children are the premises of the rule used to derive it.
- Notation: $S \vdash c$ means an argument for claim c supported by assumptions S.

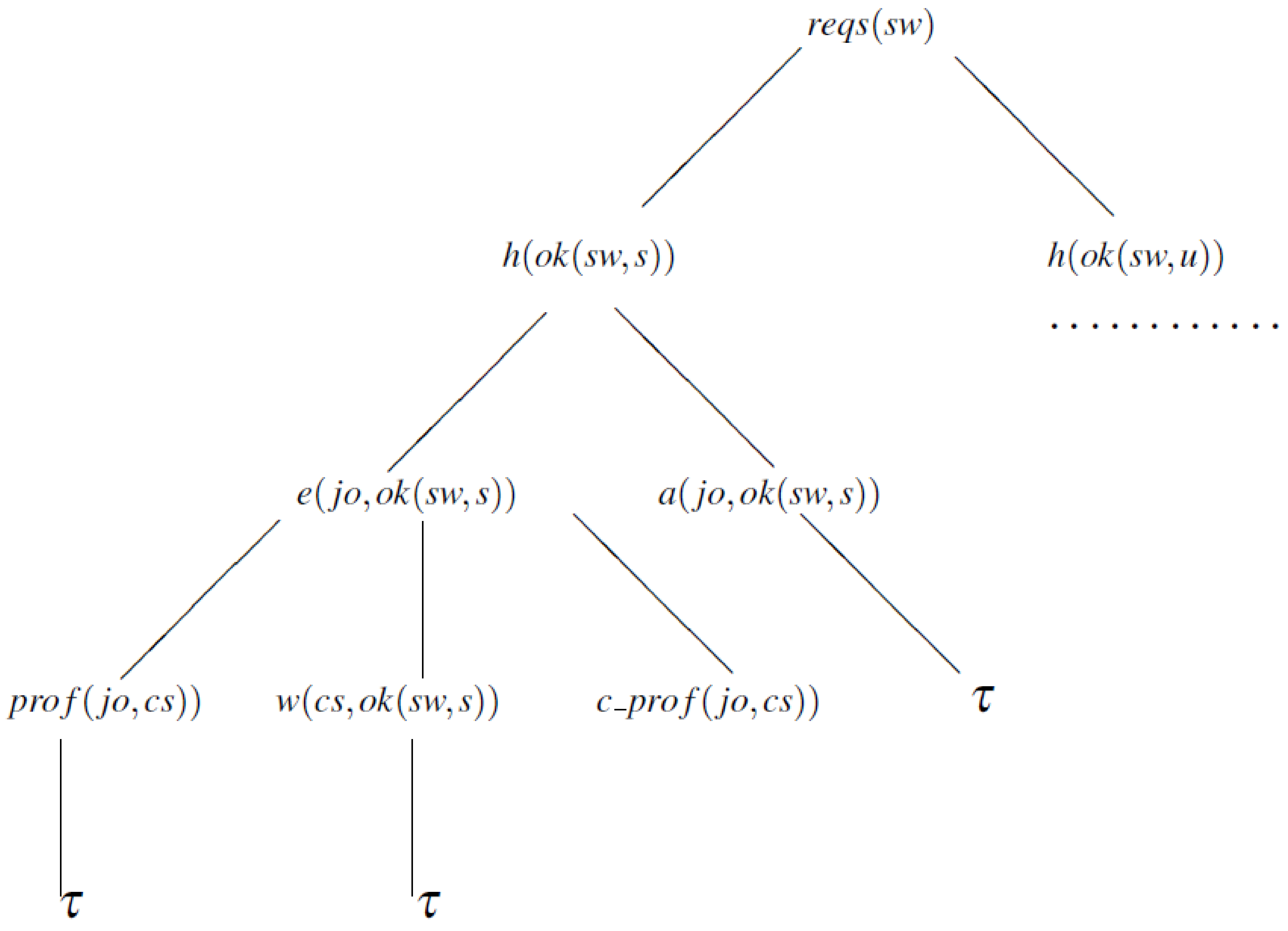

Fig. 1.1 An example argument represented as a tree

ATTACKS

One argument attacks another if the first argument's conclusion is the contrary of an assumption supporting the second argument.
This is called an "undercutting" attack.

CONTRARY

- A sentence that represents a challenge to an assumption.
- Ex. If "it will rain" is an assumption, its contrary might be "the sky is clear"

REBUTTALS TO UNDERCUTTING

- Rebuttals challenges the conclusion.
- Undercutting challenges the assumptions.
- ABA handles this by making part of the rebuttal rule defeasible. Then, an argument for the opposing conclusion can be framed as an argument for the contrary of that assumption.

- Rule 1: prog(X) ← works_for(X,micro), nor(X)
- Rule 2: ¬prog(X) ← theo(X)
    - Set contrary of nor(X) to ¬prog(X), and contrary of theo(X) to prog(X).

Definition 3. An ABA framework is a tuple $\langle \mathcal{L}, \mathcal{R}, \mathcal{A}, \overline{} \rangle$ where

- $(\mathcal{L}, \mathcal{R})$ is a deductive system, with a language $\mathcal{L}$ and a set of inference rules $\mathcal{R}$,
- $\mathcal{A} \subseteq \mathcal{L}$ is a (non-empty) set, whose elements are referred to as *assumptions*,
- $\overline{}$ is a total mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\overline{\alpha}$ is the *contrary* of α.

ACCEPTABILITY OF ARGUMENTS

- Similar to AA
  - Admissible Set
    - doesn’t attack itself, defends itself
  - Complete Set
    - admissible set, contains defended {arg}
  - Grounded Set
    - The least (smallest) complete set of {arg}

CONSISTENCY

‘If the underlying logic has negation and inconsistency, and rebuttals are reduced to undercutting, then an argument attacks itself if its support is inconsistent.’

COMPUTATION

1) Generating Arguments
2) Determining their acceptability

DISPUTE TREES

Given:
Winning Strategy as Proponent (P),
Defend an initial argument against an Opponent (O)

DISPUTE TREES

Structure:

- Nodes are arguments, labeled as P or O.
- Root is a P-node with the initial argument.
- For every P-node (argument b), and for every argument c attacking b, there's a child O-node labeled c. (P must consider all attacks).
- For every O-node (argument b), there's exactly one child P-node attacking b. (P only needs one successful counter-attack).

DISPUTE TREES

Defence Set:

- The set of all arguments in the P-nodes of a dispute tree.

Winning:

- Finite branches ending in P: P made a point O couldn't attack.
- Infinite branch: P can counter-attack every O attack indefinitely.

$$
\begin{aligned}
\mathscr{R}:\ & reqs(sw) \leftarrow h(ok(sw,s)), h(ok(sw,u)); \quad h(A) \leftarrow e(E,A), a(E,A);\\
& e(X,A) \leftarrow eng(X), w(cs,A), c\_eng(X); \quad e(X,A) \leftarrow prof(X,S), w(S,A), c\_prof(X,S);\\
& a(jo, ok(sw,s)) \leftarrow; \quad a(jo, ok(sw,u)); \quad eng(jo) \leftarrow; \quad prof(jo,cs) \leftarrow;\\
& w(cs, ok(sw,s)) \leftarrow; \quad w(cs, ok(sw,u)) \leftarrow;\\
& \neg c\_eng(E,cs) \leftarrow \neg prog(E); \quad \neg prog(X) \leftarrow theo(X);\\
& prog(X) \leftarrow works\_for(X, micro), nor(X); \quad works\_for(bob, micro) \leftarrow;\\
& \neg c\_prof(X,S) \leftarrow ret(X), inact(X); \quad ret(jo) \leftarrow;\\
& \neg c\_prof(X,S) \leftarrow admin(X), inact(X); \quad admin(jo) \leftarrow;\\
& act(X) \leftarrow pub(X); \quad pub(jo) \leftarrow\\
\mathscr{A}:\ & c\_eng(X); \quad c\_prof(X,S); \quad theo(X); \quad nor(X); \quad inact(X)\\
\overline{}:\ & \overline{c\_eng(X)} = \neg c\_eng(X); \quad \overline{c\_prof(X,S)} = \neg c\_prof(X,S);\\
& \overline{theo(X)} = prog(X); \quad \overline{nor(X)} = \neg prog(X); \quad \overline{inact(X)} = act(X)
\end{aligned}
$$

Fig. 1.3 ABA framework for the running example.

Definition 5. A dispute tree $\mathcal{T}$ is

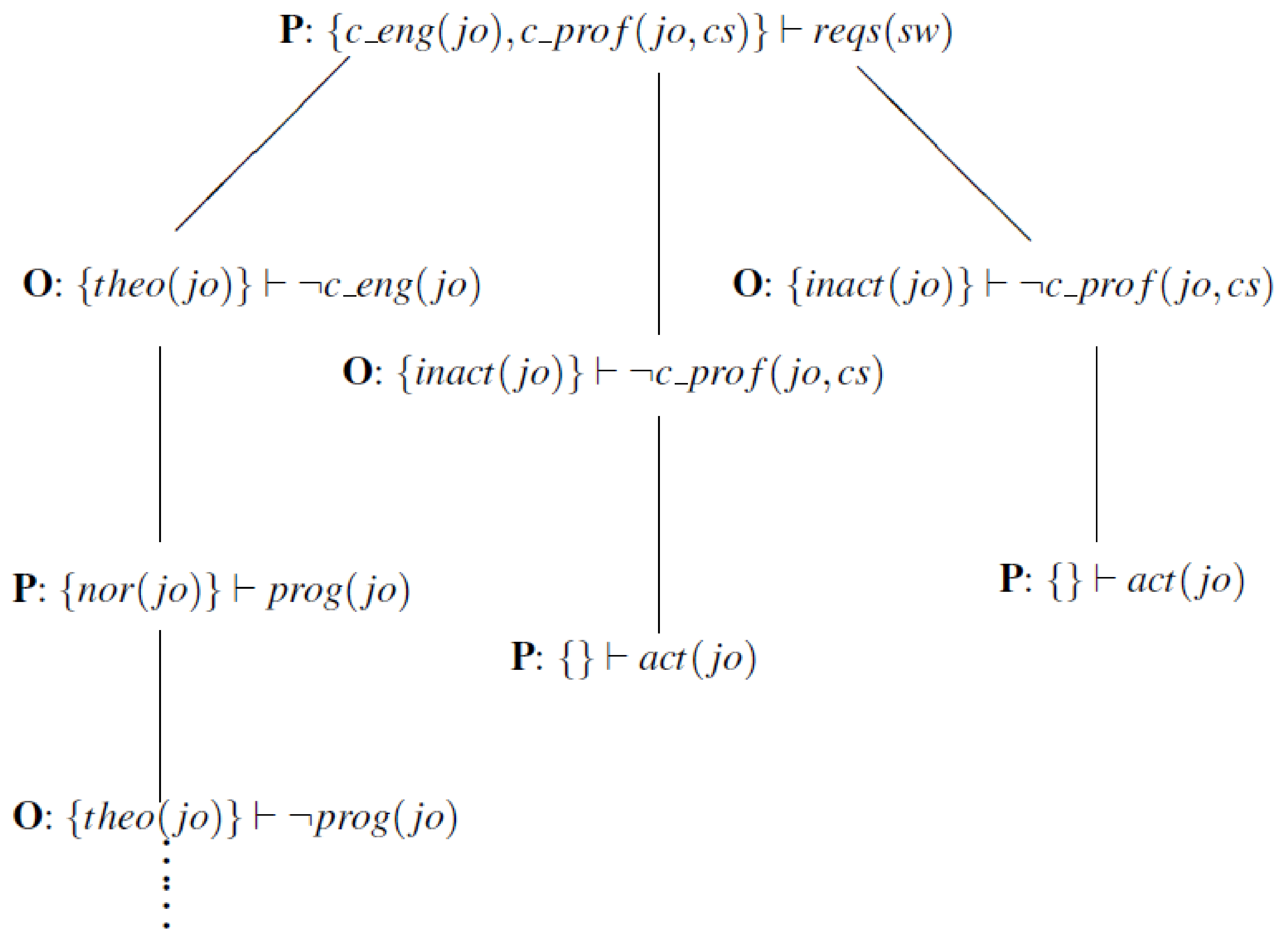

Fig. 1.2 A dispute tree for argument $\{c\_eng(jo), c\_prof(jo, cs)\} \vdash reqs(sw)$, with respect to the ABA framework in Fig. 1.3.

TYPES OF DISPUTE TREES

- **Admissible Dispute Tree**: No argument labels both a P-node and an O-node (P doesn't attack itself). Its defence set is admissible.
- **Grounded Dispute Tree**: It's finite. Its defence set is a subset of the (overall) grounded set of arguments. Any grounded dispute tree is also admissible.

DISPUTE DERIVATIONS

- Computes (approximations of) dispute trees top-down.
- Alternating between argument construction and acceptability checking (efficient).

Dung, P. M., Kowalski, R. A., & Toni, F. (n.d.). Assumption-Based Argumentation.