An Automata View to Goal-directed Methods

Lisa Hutschenreiter[1] and Rafael Peñaloza[2]

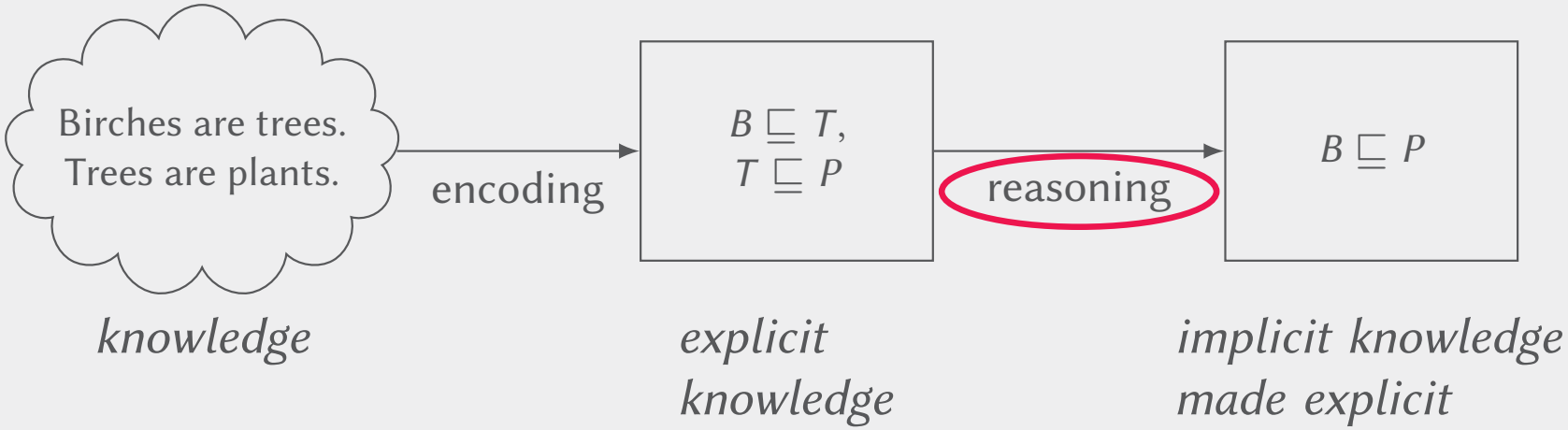
[1]Technische Universität Dresden, Germany

[2]Free University of Bozen-Bolzano, Italy

LATA 2017, Umeå, Sweden
6 March 2017

Introduction

General task:
Given some *knowledge*, find out whether some *assertion* holds.



Approach:

- encode knowledge using *suitable logic*
- make implicit knowledge available via *reasoning*

Introduction

What would the ideal reasoning method look like?

It should be

- *optimal* regarding computational complexity
- *easy* to implement
- *well-behaved* in practice
- *intuitively* understood

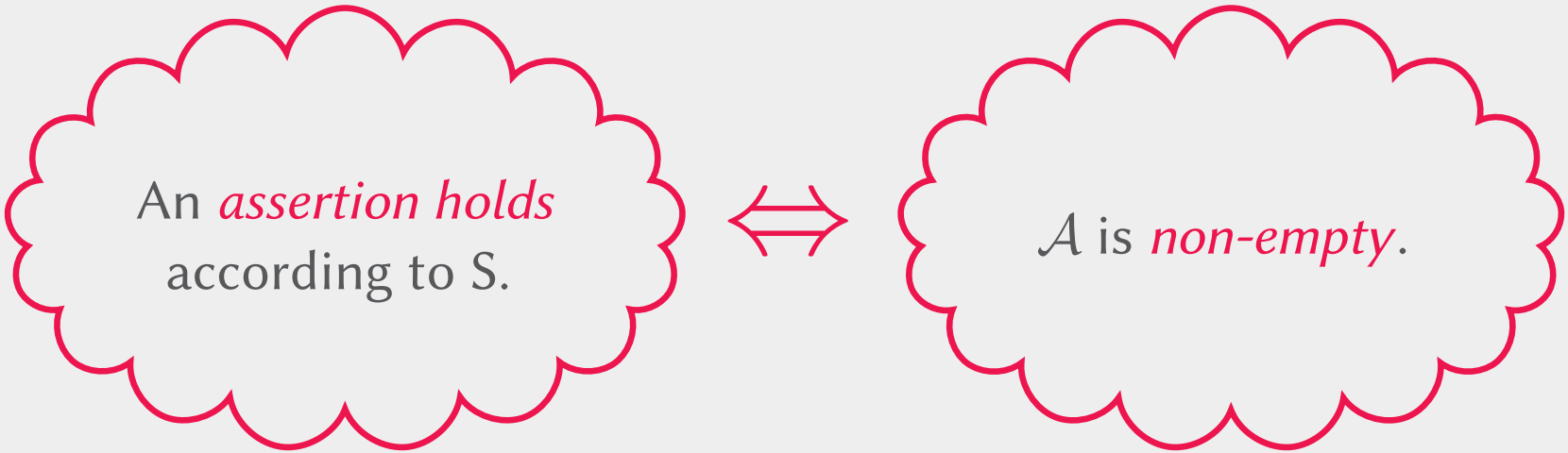
Two prominent families of approaches:

- **automata-based**
- **consequence-based**

} **combinable?**

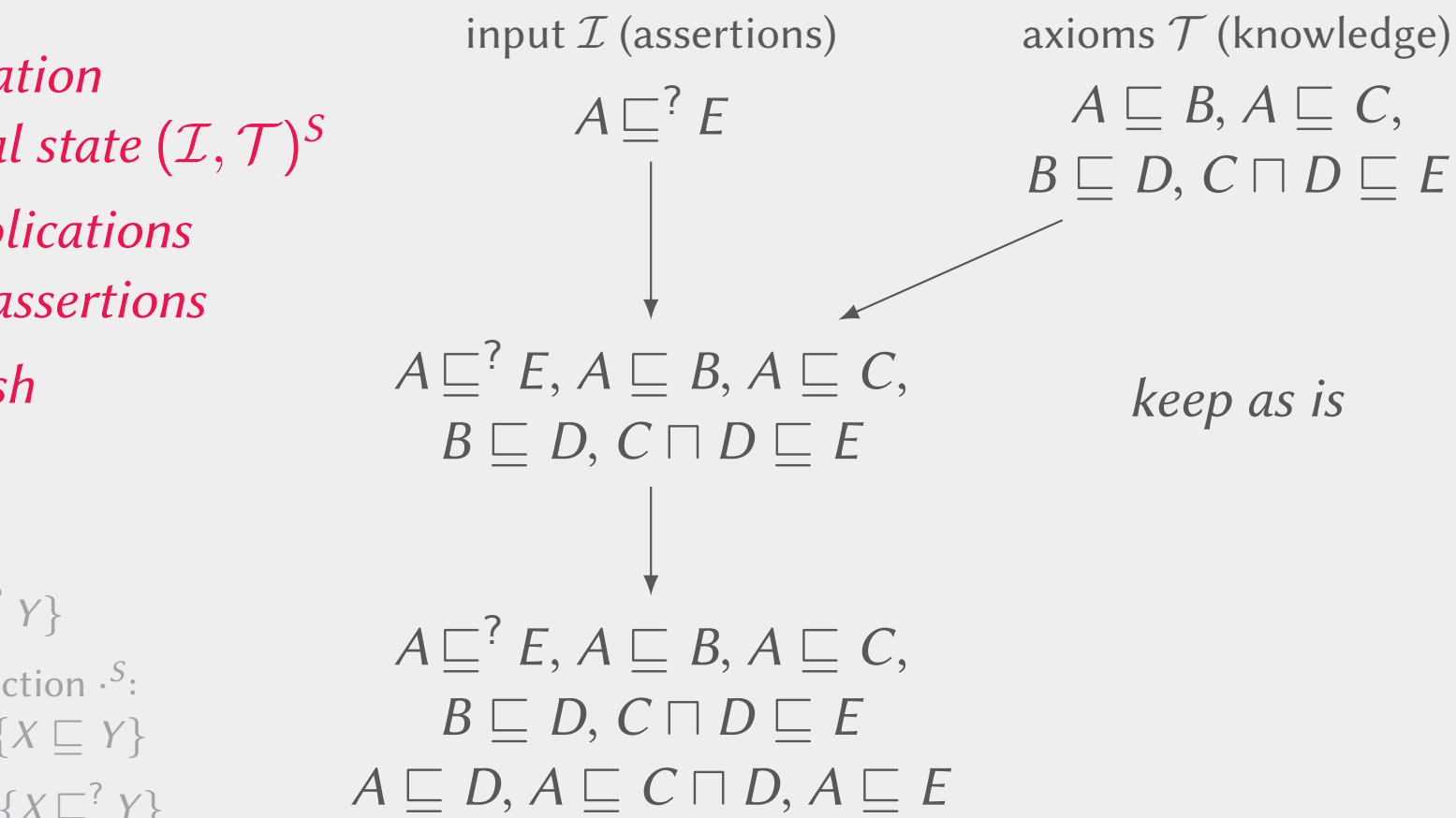
An Idea

Given a consequence-based algorithm S,
is it possible to construct an automaton $\mathcal{A}$
such that the following is true?



An Example for a Consequence-based Algorithm

- *initialisation*
  → initial state $(\mathcal{I}, \mathcal{T})^S$
- *rule applications*
  → add assertions
- *find clash*



Signature Σ:
$\{X \sqsubseteq Y, X \sqsubseteq^? Y\}$

initialisation function $\cdot^S$:
$(X \sqsubseteq Y)^S = \{X \sqsubseteq Y\}$
$(X \sqsubseteq^? Y)^S = \{X \sqsubseteq^? Y\}$

clashes $\mathcal{C}$:
$\{\{X \sqsubseteq Y, X \sqsubseteq^? Y\}\}$

rules $\mathcal{R}$:
$R_{\sqsubseteq} : (\{X \sqsubseteq Y\}, \{Y \sqsubseteq Z\}) \to \{X \sqsubseteq Z\}$
$R^+_{\sqcap} : (\{X \sqsubseteq Y, X \sqsubseteq Z\}, \emptyset) \to \{X \sqsubseteq Y \sqcap Z\}$

For an interpretation $\mathcal{J}$ it is
$C \sqsubseteq D \iff C^{\mathcal{J}} \subseteq D^{\mathcal{J}}$
$(C \sqcap D)^{\mathcal{J}} = C^{\mathcal{J}} \cap D^{\mathcal{J}}$

An Example for a Finite Tree Automaton

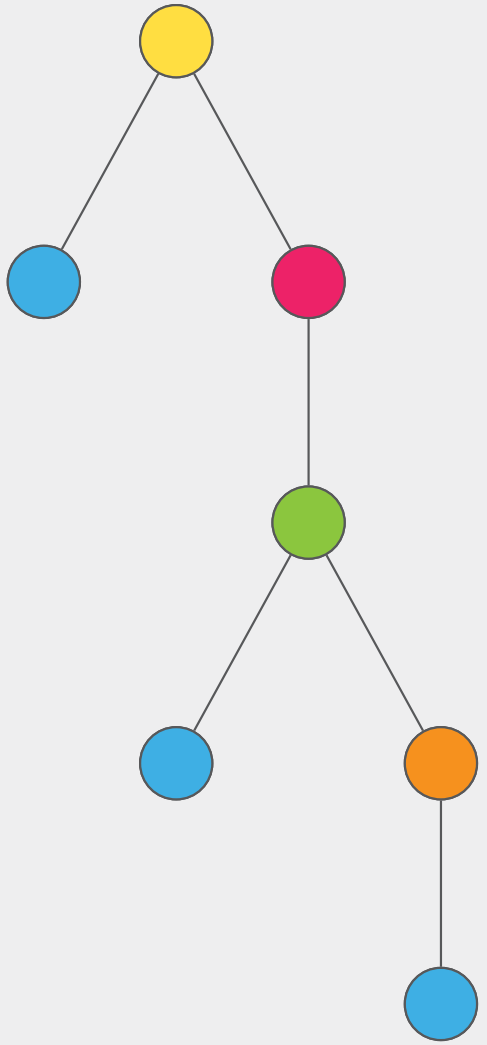
Consider the automaton $\mathcal{A} = (Q, \Delta, I, F)$:

$Q = \{$●, ●, ●, ●, ●$\}$

$I = \{$●$\}$

$F = \{$●$\}$

$\Delta = \{$(●: ●), (●: ●), (●: ●), (●: ●, ●),
(●: ●, ●), (●: ●, ●)$\}$



- *run* of $\mathcal{A}$ on a tree t:
  labelling lab: $t \to Q$ respecting Δ
- *successful* run:
  $\text{lab}(\textit{leaf}) \in F$ for all leaf nodes
- *accepting* successful run:
  $\text{lab}(\textit{root}) \in I$ for the root node

$\mathcal{A}$ is *non-empty* if there exists a finite tree t and an accepting run on t.

Why Tree Automata?

$A \sqsubseteq^? E, A \sqsubseteq B, A \sqsubseteq C,$
$B \sqsubseteq D, C \sqcap D \sqsubseteq E$

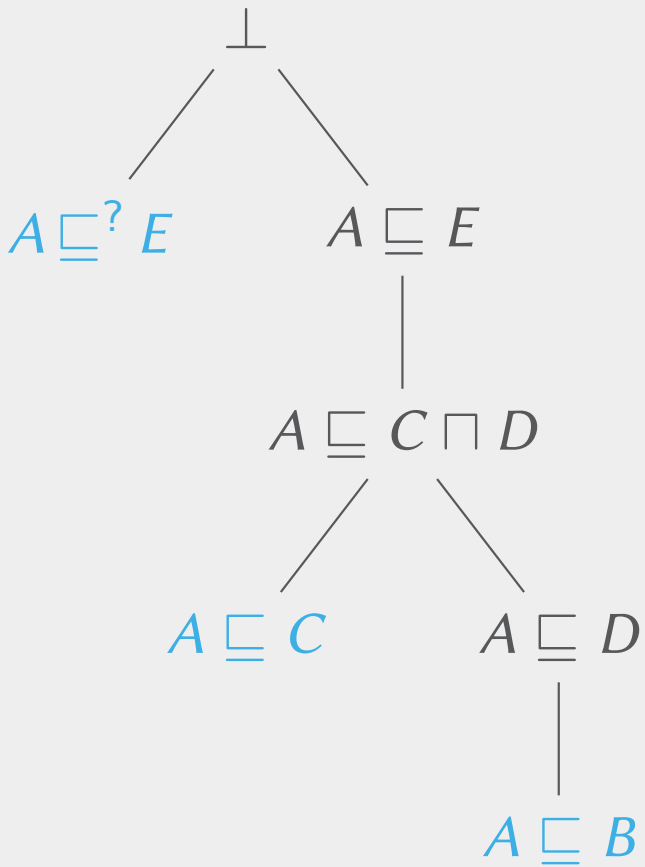
$R_{\sqsubseteq}: (\{A \sqsubseteq B\}, \{B \sqsubseteq D\}) \rightarrow \{A \sqsubseteq D\}$

$R^{+}_{\sqcap}: (\{A \sqsubseteq C, A \sqsubseteq D\}, \emptyset) \rightarrow \{A \sqsubseteq C \sqcap D\}$

$R_{\sqsubseteq}: (\{A \sqsubseteq C \sqcap D\}, \{C \sqcap D \sqsubseteq E\}) \rightarrow \{A \sqsubseteq E\}$

$A \sqsubseteq^? E, A \sqsubseteq B, A \sqsubseteq C,$
$B \sqsubseteq D, C \sqcap D \sqsubseteq E$
$A \sqsubseteq D, A \sqsubseteq C \sqcap D, A \sqsubseteq E$

clash: $\{A \sqsubseteq E, A \sqsubseteq^? E\}$



derivation tree

From Consequence-based to Tree Automata

axiomatised input

$\Gamma = (\mathcal{I}, \mathcal{T})$

Consequence-based algorithm

$S = (\Sigma, \cdot^S, \mathcal{R}, \mathcal{C})$:

- signature Σ
- initialisation function $\cdot^S$
- set of rules $\mathcal{R}$
- set of clashes $\mathcal{C}$

Tree Automaton

$\mathcal{A}_S(\Gamma) = (Q, \Delta, I, F)$:

- set of states $Q = \Sigma_\Gamma$
- set of initial states $I = \bot$
- set of final states $F = \Gamma^S$
- transition relation Δ

$$\Delta = \{\delta_R \mid R \in \mathcal{R}_\Gamma \cup \mathcal{C}_\Gamma\}$$

Our Main Result

Theorem

Let S be a consequence-based algorithm,
and $\Gamma = (\mathcal{I}, \mathcal{T})$ an axiomatised input.

Suppose that S correctly decides whether $\mathcal{I}$ follows from $\mathcal{T}$,
and consider the tree automaton $\mathcal{A}_S(\Gamma)$.

Then the following equivalence holds:

$\mathcal{I}$ *follows from* $\mathcal{T}$ according to S iff $\mathcal{A}_S(\Gamma)$ is *non-empty*.

Thank you!