MONASH University

Information Technology

# The Mobile Semantic Web

Shonali Krishnaswamy[a] & Yuan-Fang Li[b]

[a] Institute of Infocomm Research, A*STAR, Singapore
[b] Faculty of Information Technology, Monash University, Australia

{shonali.krishnaswamy,yuanfang.li}@monash.edu

# Agenda

1. Introduction & Motivation

2. A brief introduction to ontology languages & reasoning

3. Mobile Ontology Reasoning

4. Related Work & Recent Development

5. Discussions

# The Mobile Semantic Web — Why?

# Why Mobile Semantic Web ?

- **Mobile Web Services**
  - Proliferation of mobile devices
  - Service oriented paradigm + Mobile / pervasive computing
  - Increasing research focus into mobile services
- **Emerging Focus** – Mobile Hosted Web Services
  - Leverage the increasing computational capabilities of today's mobile devices to **host services**
  - **Mobile Hosted Web Services can be** accessed by both the user and other devices typically in a localised area

# Why Mobile Semantic Web ?

- Why Mobile Hosted Web Services
  - **Discovery of services in a local precinct** (e.g. Sydney Airport)

# Why Mobile Semantic Web ?

- Why Mobile Hosted Web Services
  - **Infrastructure-less" peer-to-peer** (P2P) information sharing
    - Technology Examples: http://www.scalify.com (a p2p scalable data sharing engine) – Android release in Feb 2012
    - Application Examples: Mobile/Location-based Real-Time Dating Services / Social Networking (iPhone's Grindr / Blendr Apps etc.)
  - http://m.theage.com.au/digital-life/digital-life-news/social-networking-apps-bring-the-close-closer-20111125-1nzav.html
  - Mobile Crowdsensing Applications

# Why Mobile Semantic Web ?

- Why Mobile Hosted Web Services
  - **On-device services management**
    - Google and Yahoo offer many mobile applications such as blogging, news, finance and sports
    - Apple iPhone has thousands of “apps. for everything”
    - With more and more mobile applications being published as services, the user needs a mechanism to help find the application which meets his or her particular requirements.

# Why Mobile Semantic Web ?

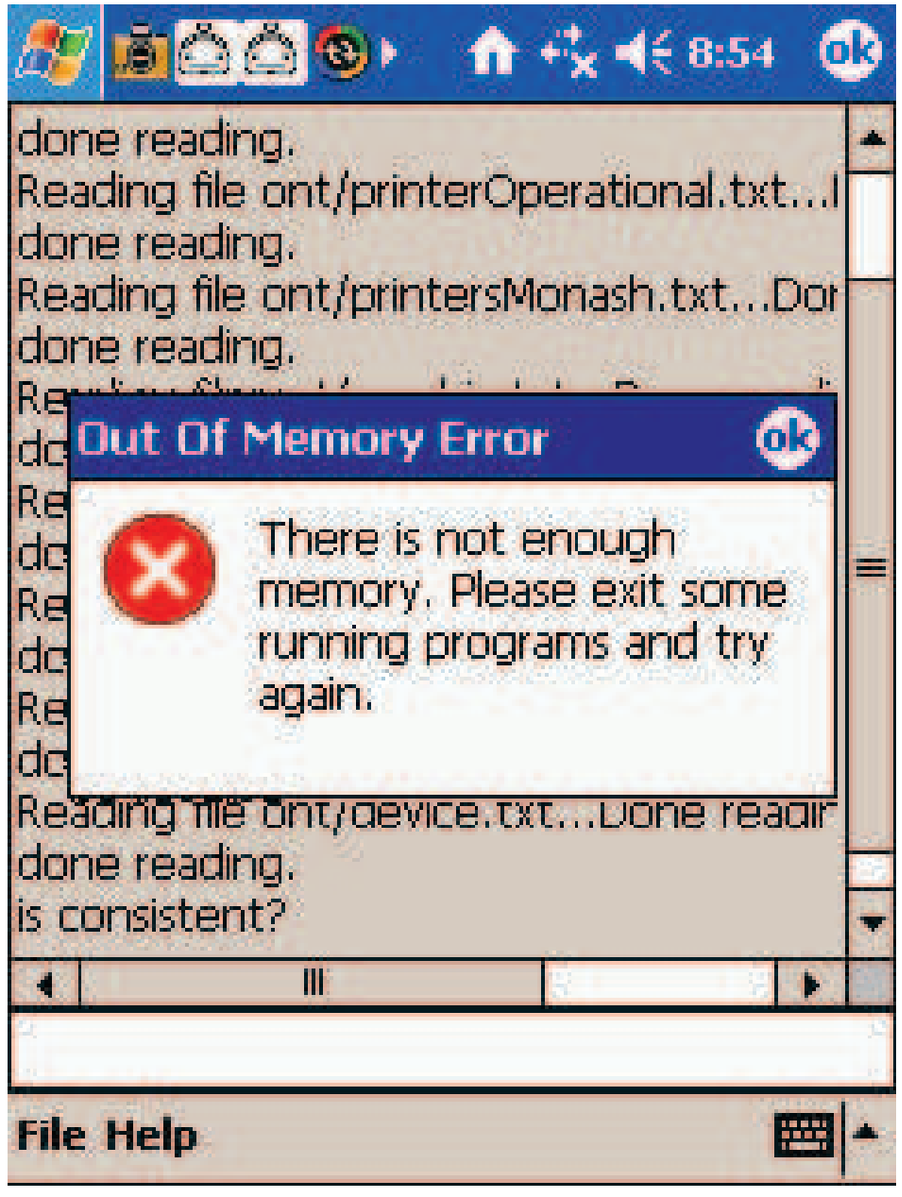
- Advantages of Mobile Hosted Web Services for the application scenarios
  - **Privacy** – Localised management of user data and applications (when needed)
  - **Battery Usage –** Continuous communication is more expensive on energy usage than processing
  - **Disconnections-** Mobile users face intermittent connectivity
  - **Network Bottlenecks/Scalability –** The Cloud can provide storage and compute resources, but mobile networks are still a bottleneck

# Mobile Hosted Web Services

- **Key Challenges**
- Identify services that are relevant to the user's changing context: *location* , *device connectivity levels* , *QoS*
- Small mobile devices are typically resource constrained in terms *of processing power, memory capabilities, screen size, battery life*
- Mobile users require quickly
  - Attention Span – Approx 15 s
- Information needs at a high level  *result accuracy*

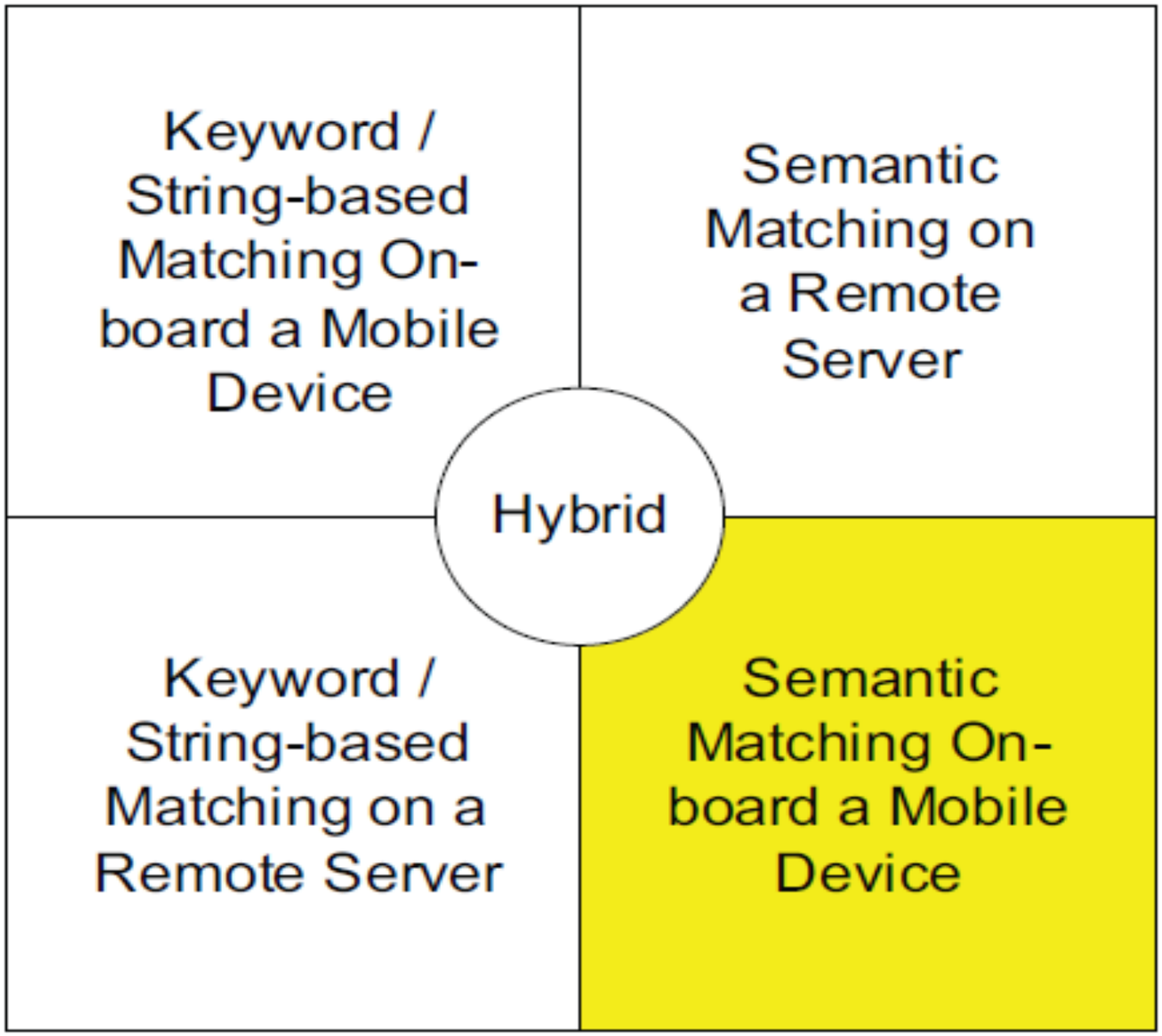
# Mobile Hosted Web Services

- **Key Challenges**
  - Fast service matching tends to be typically key-word based
  - However, lacks accuracy
  - Semantics-based matching are limited by the time and space complexity of semantic reasoning



# Mobile Service Matching — *Background*

# Service Matching in Mobile Environments



# Keyboard/Interface Matching on Mobile Device

- **Large body of existing research**

- Simple matching techniques – deployed on-board the mobile device

- Can operate in a P2P manner as well (e.g. UPnP)

- Techniques:
- String matching of keywords, Interface matching (WSDL), Extra-functional properties based (e.g QoS), convert service descriptions into a 128-bit integer using a hash function and compare these integers for service matching, location matching etc.

# Keyboard/Interface Matching on Mobile Device

- **Advantages:**
  - No remote infrastructure is required
  - Matching process occurs on-board the mobile devices
  - Uses the information gathered from other devices within network range

- **Disadvantage:**
  - Do not support semantic reasoning
    - Less accurate that semantic techniques

# Semantic Matching on Remote Servers

- **Large body of existing research**
- Key Examples:
  - CoBrA (Chen et al., 2004) was one of the early middleware architectures which utilises semantics and context to reason about users
  - Task Computing Project (TCP) (Masuoka et al., 2003) utilises OWL-S for modelling services
  - Integrated Global Pervasive Computing Framework (IGPCF) offers web service discovery using semantics on the web but assumes that pervasive users are permanently connected to the Internet.

# Semantic Matching on Remote Servers

- Luo et al. (2005, 2006) adds OWL-S descriptions to the UDDI service registry which performs inferences when a service description is published to it.

- DReggie Chakraborty et al. (2001) extends Jini to support semantic matching using Prolog for reasoning.

- LARKS (Sycara et al., 2002) is designed to match service descriptions with requests, using its own semantic description language

- The CMU Matchmaker (Srinivasan et al., 2005) provides inference based reasoning to compare OWL-S service profiles with requests, which can be stored in a back-end UDDI registry.

# Semantic Matching on Remote Servers

- Bener et al. (2009) proposes a matchmaking algorithm which also takes preconditions and effects into consideration using SWRL.

- Stuckenschmidt and Kolb (2008) defines a reasoning approach which supports partial matching of services against requests where there is insufficient time to complete the full matching process. However, this is achieved by reducing the number of conditions in the request. Reasoning is with Pellet.

- Semantic Web Engineering - Environment and Tools (SWE-ET) (Brambilla et al., 2006) combines the CEFREIEL Glue42 discovery engine with the WebRatio framework to support WSMO Semantic Web Service discovery.

# Semantic Matching on Remote Servers

- The Internet Reasoning Service (IRS)-III 46 (Domingue et al., 2008) is a Semantic Web Service broker and reasoning environment which is again based on WSMO but has
- the added functionality of importing OWL ontologies.

- DIANE (Kuster and König-Ries, 2008) is an environment for automated service discovery and matching which uses its own service profile language to describe a service as a set of effects. It supports a subset of logic without any rules or quantifiers and provides “fuzzy” matching of conditions in the user request against the service description, to provide ranked service results.

# Semantic Matching on Remote Servers – *Supporting Mobile Clients*

- Broens et al. (2004); and Doulkeridis and Vazirgiannis (2008) utilise semi-OWL and RDF documents to express service
- Baousis et al. (2008); de Andrade et al. (2007); and Chen et al. (2006), support matching of mobile services, but rely on the CMU matchmaker
- Jeon et al. (2008) matches semantically described personal preferences using OWL and SWRL roles on an external server.
- Suraci et al. (2007); and Srirama et al. (2007) support
  - OWL-S service matching on a high-end node.
- Bianchini et al. (2006) provides UDDI based matchmaking of semantically described services in a mobile environment based on location and device capabilities

# Semantic Matching on Remote Servers – *Supporting Mobile Clients*

- Veijalainen et al. (2006) performs mobile semantic service matching **on laptops which are not resource constrained**
- Wang and Hu (2008) is a P2P semantic OWL-S matching architecture which attempts to reduce the number of inference checks required.
- Niazi and Mahmoud (2009); Wolowski et al. (2007); Zoric et al. (2007a); El-Sayed and Black (2006); Almeida et al. (2006); and Sycara et al. (2002) matches services described in OWL using the Jena
- Peng et al. (2008) delegates OWL service matching to a **resource capable machine** and uses RacerPro
- AIDAS (Toninelli et al., 2008) performs matching of mobile user preferences and device capabilities for services using Pellet

# Semantic Matching on Remote Servers – *Supporting Mobile Clients*

- Gaia (Ranganathan and Campbell, 2003) is a semantically driven context mobile middleware which performs matching utilising the FaCT++ reasoner.

- Patel and Chaudhary (2009); De and Moessner (2008); and Wei et al. (2008) support semantic queries and matching by making use of Jess First Order Logic (FOL) reasoner.

- Agostini et al. (2007); Mokhtar et al. (2008); and Bouillet et al. (2008) perform all inferences offline on a high-performance server, before matching is later completed on-board a resource constrained device

  - **Hybrid Approach**

# Semantic Matching on Remote Servers – *Supporting Mobile Clients*

- All of the semantically driven approaches operate on a high-performance machine
  - require such a machine to perform pre-processing before the matching occurs.
- Reasoners Used: Jena inference prover, Jess, RacerPro, Fact++, Pellet, Prolog or Lisp.
- HermiT (Motik et al., 2009) reasoner has been developed to provide more optimised Tableaux semantic DL reasoning using OWL.
  - HermiT has been developed for the desktop / server environment.
- ***Very few perform semantic reasoning/inference on resource-constrained mobile devices***

# Semantic Matching on-board Mobile Devices

- **Chakraborty et al. (2006); Nedos et al. (2006)**
  - match services based on semantic service types defined in a hierarchy
  - However these approaches use explicit subclass relations only (such as OWL Lite)
- Do not support semantic reasoning and inference proof
- Why?
- Current open source / commercial reasoners such as Pellet, KAON2, FaCT++, RacerPro considerably resource intensive
- Cannot function on small resource constrained devices

# Ontology Reasoning

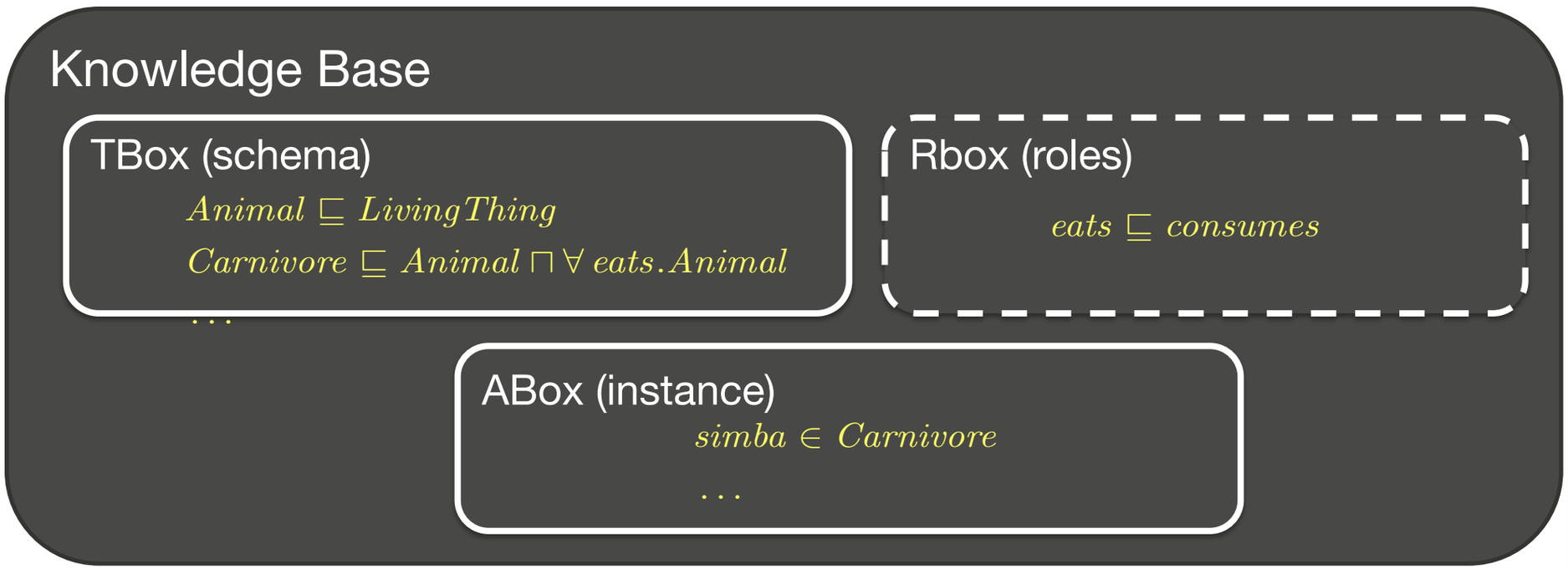
Basic Description Logics

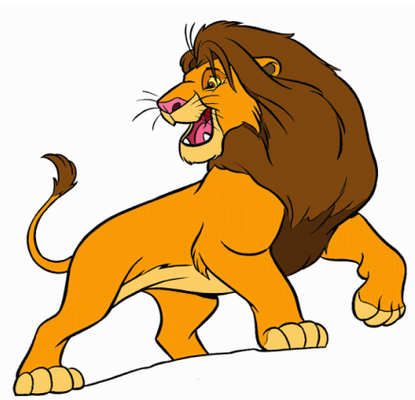
Completion Rules-based Algorithm

Tableau Algorithms

# Description Logics

- A family of knowledge representation formalisms (Baader et al (2003))
  - Successor of semantic networks & Minsky frames
  - Describes domains using concepts, roles & individuals
- Formal semantics
  - A decidable fragment of first-order logic (generally speaking)
  - Some DLs are variants of modal logic



© Disney Inc.

# Description Logics: Main Entities

- Used for *knowledge representation*
- Three types of main entities
  - **Concepts** (classes): *first-class citizen*, representing sets of abstract entities

$$MeatyPizza \sqsubseteq Pizza$$

  - **Roles** (properties, predicates): binary relations

$$hasTopping \sqsubseteq hasIngredient$$

  - **Individuals**: entities

$$Country \equiv \{America, England, France, Germany, Italy\}$$

# Description Logics: Expressions & Axioms

- Expressions used to construct complex concepts & roles

$$Food \sqcap \exists hasBase.PizzaBase$$

$$Pizza \sqcap \exists hasTopping.MeatTopping$$

- Axioms used to express relationship between concepts or roles
  - Subsumption, equivalence, disjointness

$$Pizza \sqsubseteq Food \sqcap \exists hasBase.SomeBase$$

$$MeatyPizza \equiv Pizza \sqcap \exists hasTopping.MeatTopping$$

$$VegetarianPizza \sqcap NonVegetarianPizza \sqsubseteq \bot$$

# Description Logics: Syntax

- Prototypical DL: **A**ttributive Concept **L**anguage with **C**omplements ($\mathcal{ALC}$)
  - Schmidt-Schauß & Smolka (1991)
  - Concept expressions:
    - $C ::= \top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$
  - Axioms
    - Terminological (TBox)
      - General concept inclusion (GCI): $C \sqsubseteq D,\ C \equiv D$
    - Assertional (ABox)
      - Concept assertions: $a : C$
      - Role assertions: $(a, b) : R$

# Description Logics: Semantics

- Usually specified with an *interpretation*-based semantics $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$
  - The domain of interpretation $\Delta^{\mathcal{I}}$: the universal set of individuals
  - The interpreting function $\cdot^{\mathcal{I}}$: maps concepts/roles/individuals into $\Delta^{\mathcal{I}}$
  - For $\mathcal{ALC}$

| Concept expression | Interpretation |
|---|---|
| $\top$ | $\Delta^{\mathcal{I}}$ |
| $\bot$ | $\emptyset$ |
| $A$ | $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ |
| $C_1 \sqcap C_2$ | $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ |
| $C_1 \sqcup C_2$ | $C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$ |
| $\neg C$ | $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ |
| $\exists R.C$ | $\{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ |
| $\forall R.C$ | $\{x \mid \exists y.\langle x, y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ |

# Description Logics: Reasoning Problems

- Based on the interpretation, for TBox & Abox
- Given a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$
- For TBox $\mathcal{T}$
  - Concept satisfiability $\mathcal{I} \models C$ iff $C^{\mathcal{I}} \neq \emptyset$
  - Subsumption $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
  - TBox consistency $\mathcal{I} \models \mathcal{T}$ iff $\mathcal{I} \models \Phi$, for every $\Phi \in \mathcal{T}$
- For ABox $\mathcal{A}$
  - Concept assertions $\mathcal{I} \models a\colon C$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$
  - Role assertions $\mathcal{I} \models (a, b)\colon R$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
  - ABox consistency $\mathcal{I} \models \mathcal{A}$ iff $\mathcal{I} \models \varphi$, for every $\varphi \in \mathcal{A}$
- Can all be reduced to ABox consistency

# Extensions of $\mathcal{ALC}$

- Add expressivity in different ways
- Increases reasoning complexity
- Extended DLs are composed by combining language features

| Constructor | Extension | Example |
|---|---|---|
| $\mathcal{F}$ | Functionality | $\leq 1\ R$ |
| $\mathcal{N}, \mathcal{Q}$ | (Un)qualified number restrictions | $\leq n.C,\ \geq m$ |
| $\mathcal{O}$ | Nominals (enumeration) | $\{a_1, a_2, \ldots, a_n\}$ |
| $\mathcal{S}$ | Transitive roles | $Tr(R)$ |
| $\mathcal{H}$ | Role hierarchy | $R \sqsubseteq S$ |
| $\mathcal{I}$ | Inverse roles | $R^-$ |
| $\mathcal{R}$ | Complex role inclusion | $R \circ S \sqsubseteq S$ |

# A Spectrum of Expressivity (1)

- More expressive extensions
  - OWL Lite: $\mathcal{SHIF}(\mathbf{D})$ (Horrocks & Patel-Schneider (2003))
    - Transitive, inverse, functional roles & role hierarchy,
  - OWL DL: $\mathcal{SHOIN}(\mathbf{D})$ (Horrocks, Kutz, Sattler (2006))
    - Nominals & qualified number restrictions
  - OWL 2 DL: $\mathcal{SROIQ}(\mathbf{D})$ (Horrocks, Kutz, Sattler (2006))
    - Complex role inclusions

| DL | Example |
|---|---|
| OWL Lite | $hasTopping \sqsubseteq hasIngedient$ |
| OWL DL | $\geq 3\ hasTopping$ |
| OWL 2 DL | $\geq 3\ hasTopping.MeatTopping$ |

# A Spectrum of Expressivity (2)

- Less expressive subsets
  - OWL 2 EL: $\mathcal{EL}^{++}$ (Baader, Brandt, Lutz (2005))
    - $C ::= \top \mid \bot \mid \{a\} \mid C \sqcap D \mid \exists R.C$
    - Designed for efficient reasoning over large Tboxes
      - Especially biomedical ontologies
  - OWL 2 QL: $DL\text{-}Lite_{\mathcal{R}}$ (Calvanese et al (2005))
    - $A \mid \exists R.\top \sqsubseteq A \mid C \sqcap D \mid \neg C \mid \exists R.C$
    - Designed for *very* efficient query answering over large ABoxes
  - OWL 2 RL: $DLP \ \& \ pD^*$ (Grosof et al (2003); Horst (2005))
    - Syntactic restrictions on class expressions
    - Designed for efficient rule-based reasoning

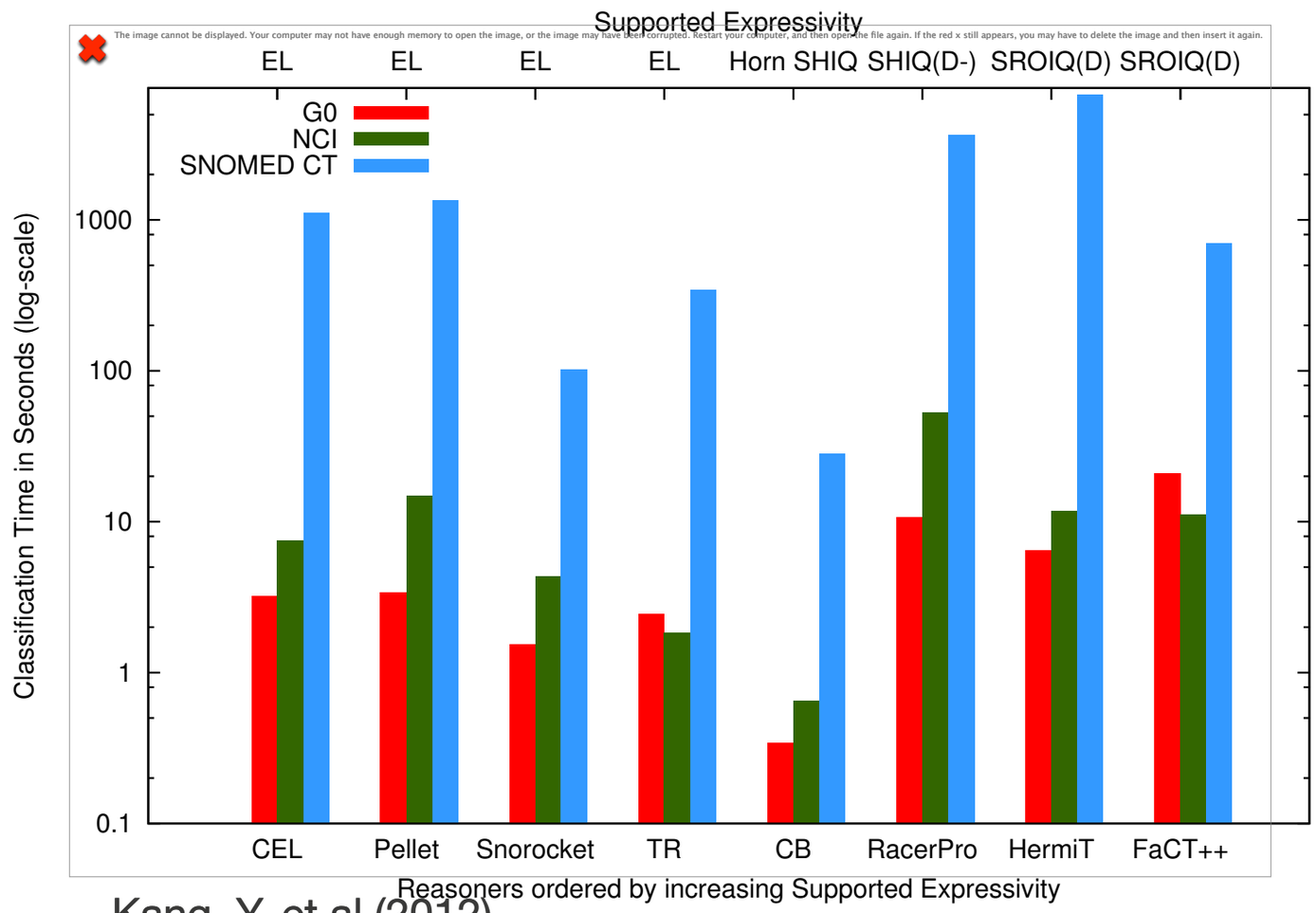
# Reasoning Complexity

- "With great expressivity comes great complexity"

| DL | | TBox consistency |
|---|---|---|
| $\mathcal{ALC}$ | | PSPACE-complete |
| $\mathcal{SHIF}(\mathbf{D})$ | (OWL Lite) | EXPTIME-complete |
| $\mathcal{SHOIN}(\mathbf{D})$ | (OWL DL) | NEXPTIME-complete |
| $\mathcal{SROIQ}(\mathbf{D})$ | (OWL 2 DL) | 2NEXPTIME-complete |
| $\mathcal{EL}^{++}$ | (OWL 2 EL) | PTIME-complete |
| $DL\text{-}Lite_{\mathcal{R}}$ | (OWL 2 QL) | NLOGSPACE-complete |
| OWL 2 RL | | PTIME-Complete |

# Inference is **Hard**

- Especially for large ontologies and/or **resource-constrained devices**




Kang, Y. et al (2012)

Dentler, K. et al (2011)

# Inference is Hard

| DL | | TBox consistency |
|---|---|---|
| $\mathcal{ALC}$ | | PSPACE-complete |
| $\mathcal{SHIF}(\mathbf{D})$ | (OWL Lite) | EXPTIME-complete |
| $\mathcal{SHOIN}(\mathbf{D})$ | (OWL DL) | NEXPTIME-complete |
| $\mathcal{SROIQ}(\mathbf{D})$ | (OWL 2 DL) | 2NEXPTIME-complete |
| $\mathcal{EL}^{++}$ | (OWL 2 EL) | PTIME-complete |
| $DL\text{-}Lite_{\mathcal{R}}$ | (OWL 2 QL) | NLOGSPACE-complete |
| OWL 2 RL | | PTIME-Complete |

- Difficulty a result of:
  - Language constructs used (union, at most, *etc.*)
  - Tableau algorithms: exhaustive exploration until all ABoxes checked
    - Complete, but **expensive**
  - Optimisation techniques employed

# Reasoning Paradigms

- Completion rules-based algorithms (Baader, Brandt, Lutz (2005))
  - Suitable for less expressive DLs: the $\mathcal{EL}$ family
  - Reasoners: CEL (Baader, Lutz & Suntisrivaraporn (2006)), REL (Ren & Pan (2010)), Snorocket (Lawley & Bousquet (2010))
- Consequence-driven algorithms (Kazakov (2009)
  - A recent, efficient algorithm for Horn $\mathcal{SHIQ}$ and beyond
  - Reasoners: ELK (Kazakov Krötzsch & Simančík (2011)), CB, ConDOR (Simančík, Kazakov, Horrocks (2011))
- Tableau algorithms (Baader & Sattler (2001))
  - Mostly widely used, suitable for very expressive DLs
  - Reasoners: FaCT++ (Tsarkov & Horrocks (2006)), Racer (Haarslev & Möller (2001)), HermiT (Shearer, Motik & Horrocks (2008)), Pellet (Sirin et al (2007))

# The Case for Less Expressive DLs

- Many large biomedical ontologies don't use/need the full expressivity of OWL 2 DL:
  - Gene Ontology, NCI Thesaurus, Gazetteer, and many, many more
  - They can be expressed using a very limited set of constructs
- $\mathcal{EL}$ was designed to exploit this fact:
  - Simpler logic, faster reasoning!
- Completion rules-based algorithm
  - An algorithm for calculating classification
    - The subsumption hierarchy of an ontology
  - Apply *completion rules* to saturate the subsumption graph
  - PTIME-complete, optimal!

# Completion Rules-based Algorithm

- Recall, in $\mathcal{EL}$
  - $C ::= \top \mid \bot \mid \{a\} \mid C \sqcap D \mid \exists R.C$
- Completion rules-based algorithm
  - An algorithm for calculating classification
    - The subsumption hierarchy of an ontology
  - Use *completion rules* to saturate the subsumption graph
- Given a TBox $\mathcal{T}$
  1. Normalise axioms in $\mathcal{T}$
  2. Map the normalised TBox to completion sets
  3. Apply completion rules to the completion sets
  4. Read subsumption relationships

# Completion Rules: Normalisation

- Apply normalisation rules to
  - Normalise GCIs into:
    - $A_1 \sqsubseteq B_1 \mid A_1 \sqcap A_2 \sqsubseteq B \mid A_1 \sqsubseteq \exists r.A_2 \mid \exists r.A_1 \sqsubseteq B \mid$
  - Normalise role inclusions into:
    - $r \sqsubseteq s \mid r_1 \circ r_2 \sqsubseteq s$
- Normalisation rules

$$
\begin{aligned}
C \equiv D &\rightarrow C \sqsubseteq D, D \sqsubseteq C \\
C^{\perp} \sqsubseteq \mathrm{D} &\rightarrow \\
C \sqsubseteq D^{\perp} &\rightarrow C \sqsubseteq \perp \\
C_1 \sqcap \ldots \hat{C} \sqcap \ldots \sqcap C_n \sqsubseteq \mathrm{D} &\rightarrow \hat{C} \sqsubseteq A, C_1 \sqcap \ldots \sqcap A \sqcap \ldots \sqcap C_n \sqsubseteq D \\
\hat{C} \sqsubseteq \hat{D} &\rightarrow \hat{C} \sqsubseteq A, A \sqsubseteq \hat{D} \\
\ldots &
\end{aligned}
$$

# Completion Rules: Completion Rules

- Two completion sets: $S$ & $R$

$$B \in S(A) \Rightarrow \mathcal{O} \models A \sqsubseteq B$$

$$(A, B) \in R(r) \Rightarrow \mathcal{O} \models A \sqsubseteq \exists r.B$$

- Completion rules

$$\frac{A_1, \ldots, A_n \in S(X), \sqcap(A_1, \ldots, A_n) \sqsubseteq B,\ B \notin S(X)}{S(X) := S(X) \cup \{B\}}\ \textbf{CR1}$$

$$\frac{A \in S(X),\ A \sqsubseteq \exists r.B \in \mathcal{T},\ (X, B) \notin R(r)}{R(r) := R(r) \cup \{(X, B)\}}\ \textbf{CR2}$$

$$\ldots$$

$$\frac{(X, Y) \in R(r),\ (Y, Z) \in R(s),\ r \circ s \sqsubset t \in \mathcal{T},\ (X, Z) \notin R(t)}{R(t) := R(t) \cup \{(X, Z)\}}\ \textbf{CR6}$$

# Tableau Algorithms

- Foundation of several highly optimised reasoners
  - FaCT++, Racer, Pellet, HermiT, *etc.*
- Support TBox reasoning problems
  - Reduction to *ABox consistency*
  - Concept satisfiability: build a *tree-like model* for $C$ by applying expansion rules on an ABox: $\mathcal{A} \models C(a)$ iff $\mathcal{A} \cup \{\neg\, C(a)\}$ is inconsistent
    - Models are ABoxes
- Tableau algorithms are *sound* & *complete*
- An ABox is
  - Complete: if no more rules apply
  - Closed: if it contains a *clash*
  - Open: if it doesn't contain a clash

# Tableau Algorithm: Basic Idea

- Basic idea: given a concept $C$
  - Apply *expansion rules* repeatedly to syntactically decompose $C$
  - Stop when
    - A *clash* occurs: $C$ is unsatisfiable, or
    - No more rules apply & no clash detected: $C$ is satisfiable
- Clash (contradiction) — many forms
  - $\{C(a), \neg\, C(a)\} \in \mathcal{A}$
  - $\{\bot(a)\} \in \mathcal{A}$
  - $\{(\leq n\ r)(a)\} \cup \{r(a, b_i) | 1 \leq i \leq n+1\} \cup \{y_i \neq y_j | 1 \leq i <\leq n+1\} \subseteq \mathcal{A}$
    for $a, y_1, \ldots, y_n + 1 \in N_I, r \in N_R, n \in \mathbb{N}$
  - ...
- Transformation rules
  - Can be deterministic or nondeterministic
  - Source of complexity

# Tableau Expansion

- Certain rules create ABoxes – all of which need to be explored!
- Expansion rules are applied in an **expansion tree**

**Algorithm 1** TableauxTreeTraverse($\mathcal{A}$, $\mathcal{G}$)

**Inputs:** ABOX $\mathcal{A}$, Expansion tree $\mathcal{G}$

**Output:** Boolean allExpansionsClash

```
1:  let ClassConcept clash ← null
2:  while moreTableauxRulesToApply(A, G) = true do
3:      let clash ← ApplyTaxleauxRules(A, G)
4:      if clash ≠ null then
5:          let openBranchFound ← BackJump(clash, A, G)
6:          if openBranceFound = false then
7:              return true
8:          end if
9:      end if
10: end while
11: return false
```

# Expansion Rules

- Each language construct has a rule: $\sqcap, \sqcup, \exists, \forall, \leq, \geq, etc.$
  - Consistency preserving
  - Can generator new individuals $\exists$-rule $\geq$-rule
  - Can be **non-deterministic** $\sqcup$-rule $\leq$-rule
  - Can be applied simultaneously – order matters greatly!

$\sqcap$-rule

$$(C_1 \sqcap C_2)(a) \in \mathcal{A}$$
$$C_1(a) \notin \mathcal{A} \wedge C_2(a) \notin \mathcal{A}$$

---

$$\mathcal{A}' := \mathcal{A} \cup \{C_1(a), C_2(a)\}$$

$\sqcup$-rule

$$(C_1 \sqcup C_2)(a) \in \mathcal{A}$$
$$C_1(a) \notin \mathcal{A} \wedge C_2(a) \notin \mathcal{A}$$

---

$$\mathcal{A}' := \mathcal{A} \cup \{C_1(a)\}, \mathcal{A}'' := \mathcal{A} \cup \{C_2(a)\}$$

$\exists$-rule

$$(\exists\, r.C)(a) \in \mathcal{A}$$
$$\nexists\, z \in \mathcal{A} \bullet C(z) \in \mathcal{A} \wedge r(a, z) \in \mathcal{A}$$

---

$$\mathcal{A}' := \mathcal{A} \cup \{C(b), r(a, b)\}$$

$\forall$-rule

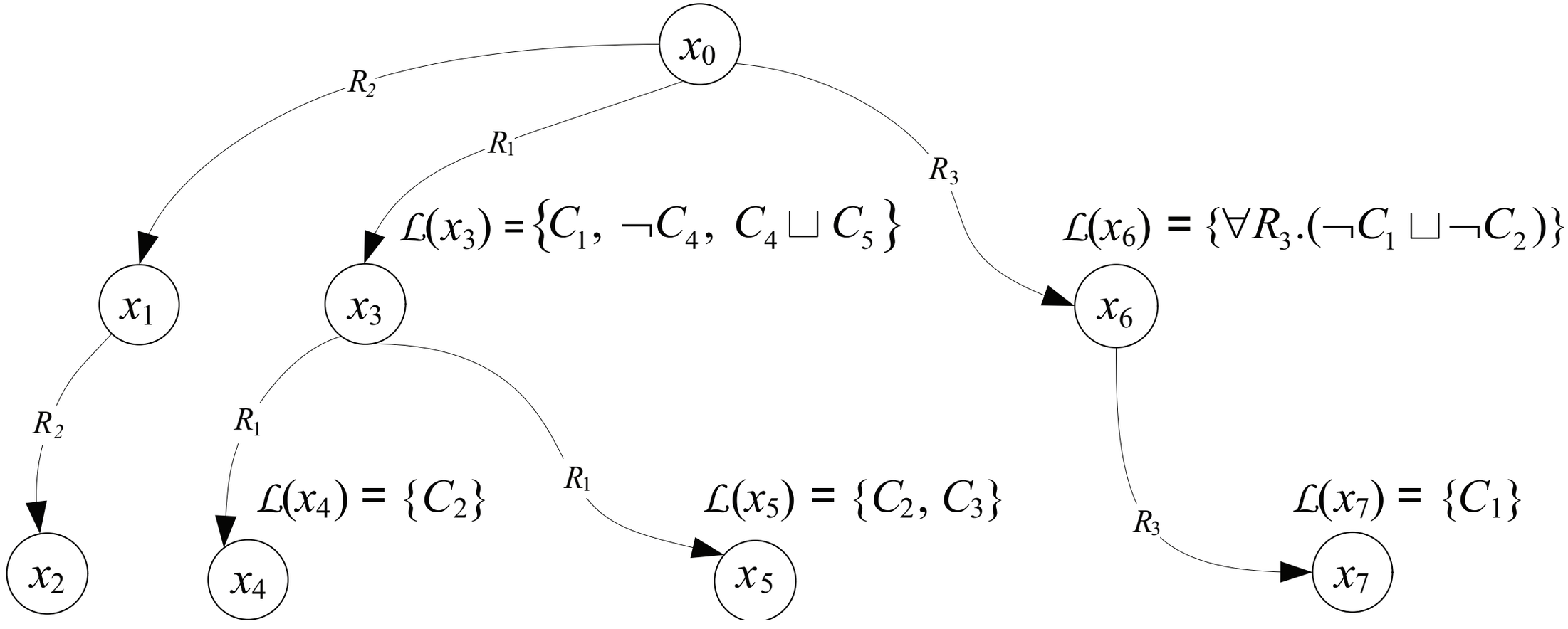
$$(\forall\, r.C)(a) \in \mathcal{A}, r(a, b) \in \mathcal{A}$$
$$C(b) \notin \mathcal{A}$$

---

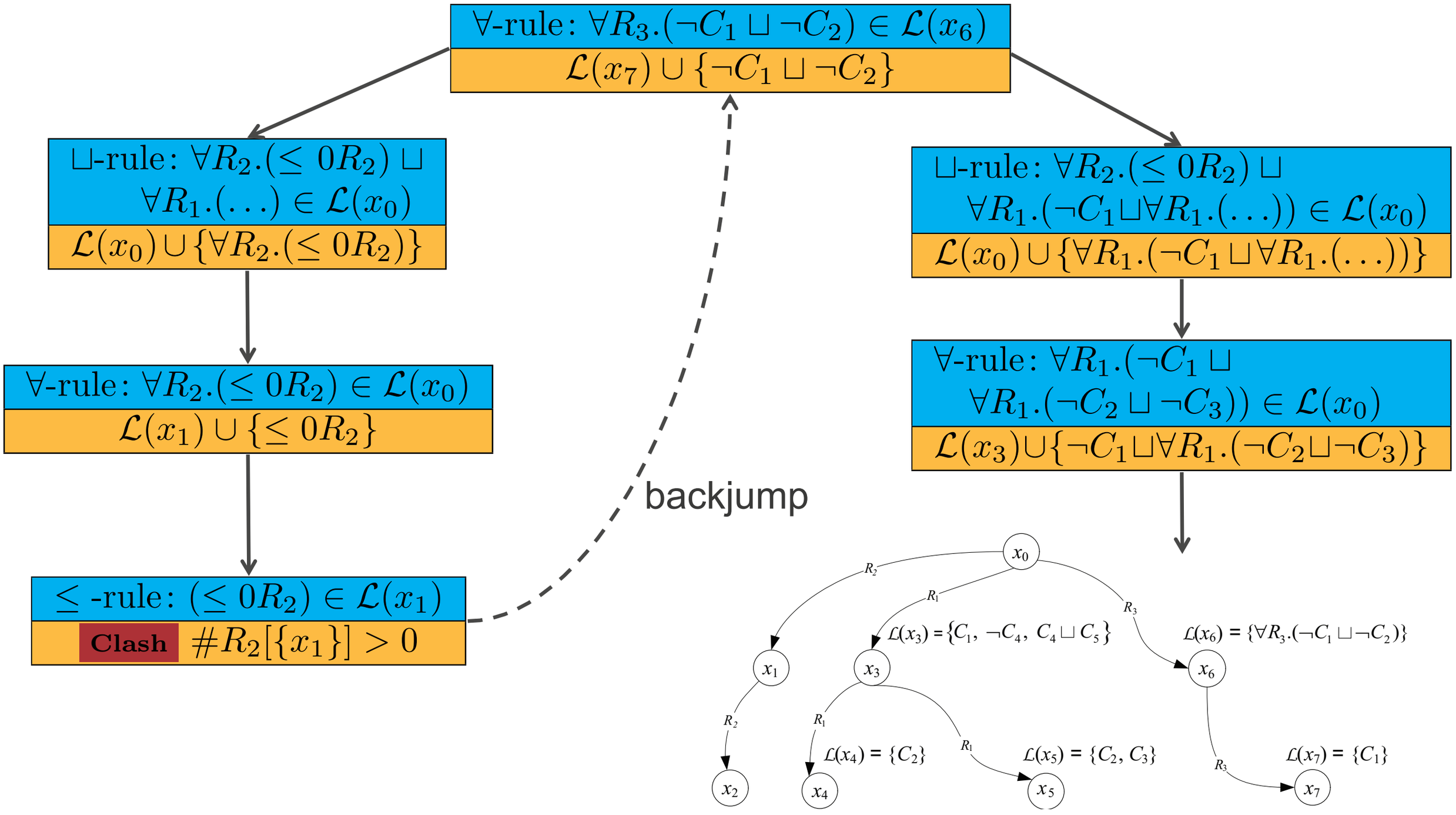
$$\mathcal{A}' := \mathcal{A} \cup \{C(b)\}$$

# Tableau Algorithm: An Example

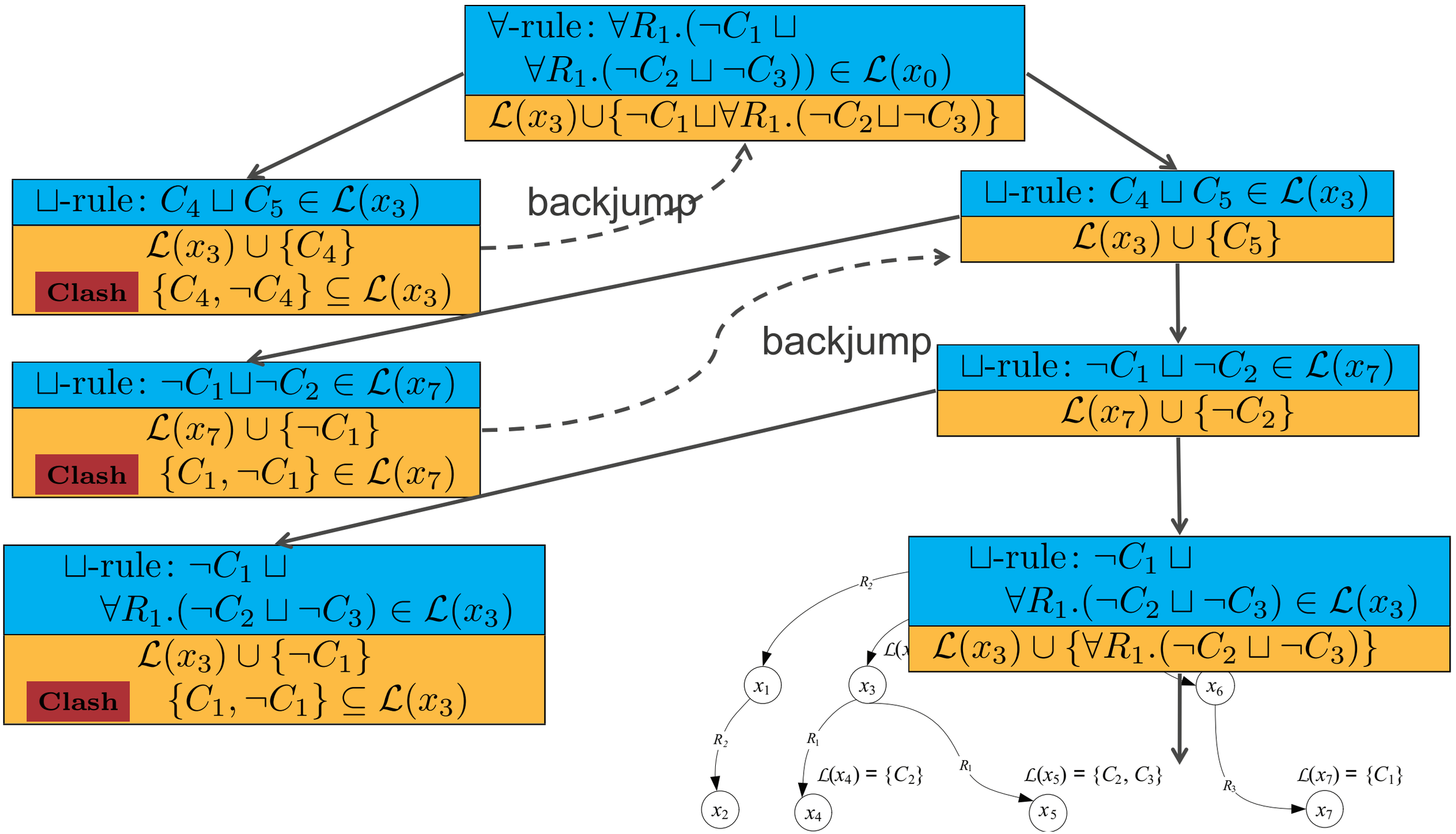
- Individuals $\{x_0, x_1, \ldots, x_7\} \subseteq \mathcal{A}$
- Assertions $\mathcal{L}(x_3) = \{C_1, \neg C_4, C_4 \sqcup C_5\}$ $\mathcal{L}(x_4) = \{C_2\}$ $\mathcal{L}(x_5) = \{C_2, C_3\}$
  $\mathcal{L}(x_6) = \{\forall R_3.(\neg C_1 \sqcup \neg C_2)\}$ $\mathcal{L}(x_7) = \{C_1\}$
- **Task** $\mathcal{A} \models C_0(x_0),$ where $C_0 \equiv \exists R_2.(\geq 1 R_2) \sqcap \exists R_1.(C_1 \sqcap \exists R_1.(C_2 \sqcap C_3))$
- **Negated** $\mathcal{A} \cup ((\forall R_2.(\leq 0 R_2)) \sqcup (\forall R_1.(\neg C_1 \sqcup \forall R_1.(\neg C_2 \sqcup \neg C_3)))(x_0)$ is closed?



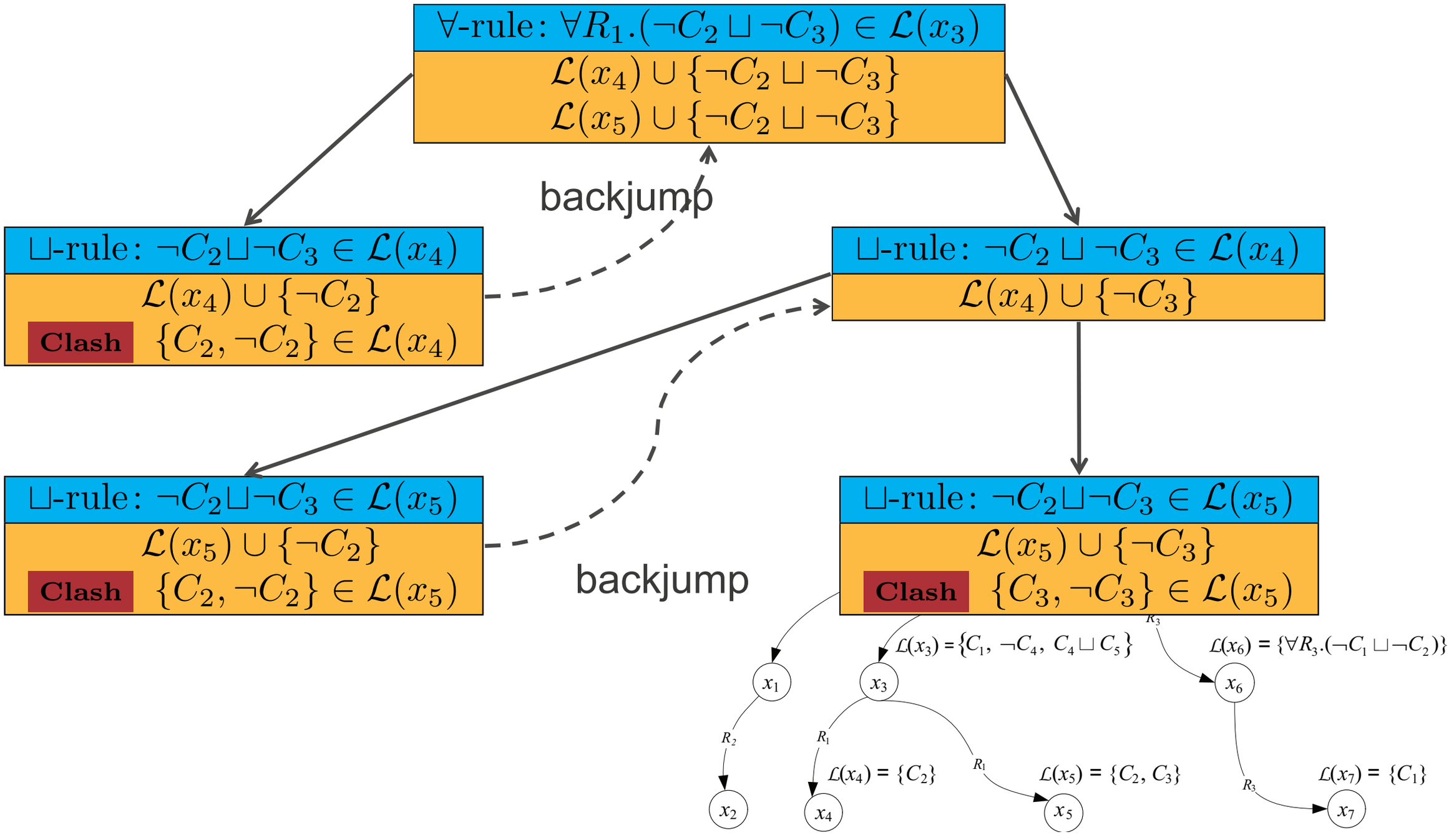
# Tableau Algorithm: An Example



# Tableau Algorithm: An Example



# Tableau Algorithm: An Example



# Observations

- Many transformation rules do not contribute to the inference problem
- Especially disjunctions

$\forall$-rule: $\forall R_3.(\neg C_1 \sqcup \neg C_2) \in \mathcal{L}(x_6)$

$\sqcup$-rule: $C_4 \sqcup C_5 \in \mathcal{L}(x_3)$

$\sqcup$-rule: $\neg C_2 \sqcup \neg C_3 \in \mathcal{L}(x_4)$

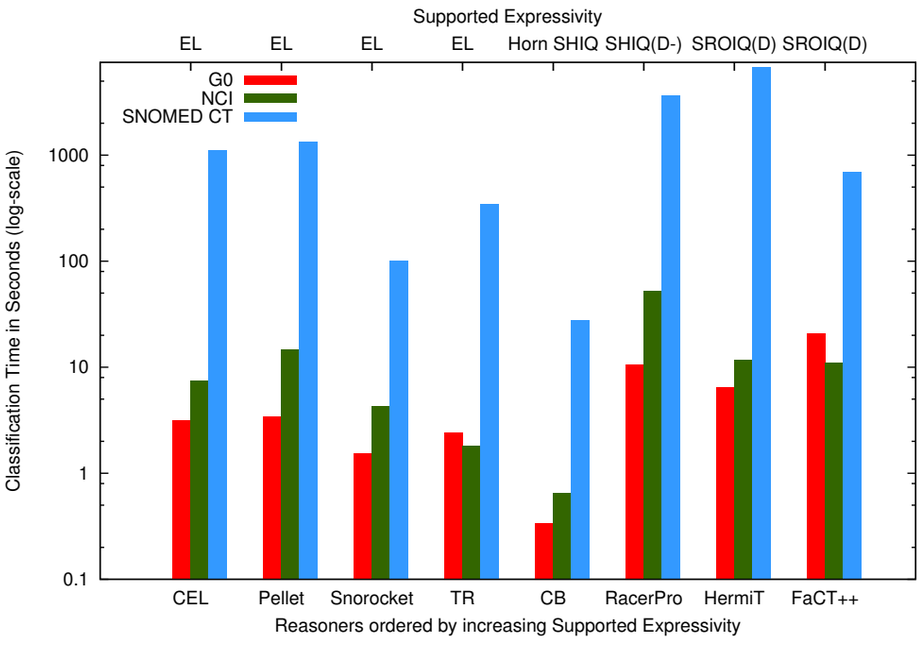
$\sqcup$-rule: $\neg C_1 \sqcup \neg C_2 \in \mathcal{L}(x_7)$

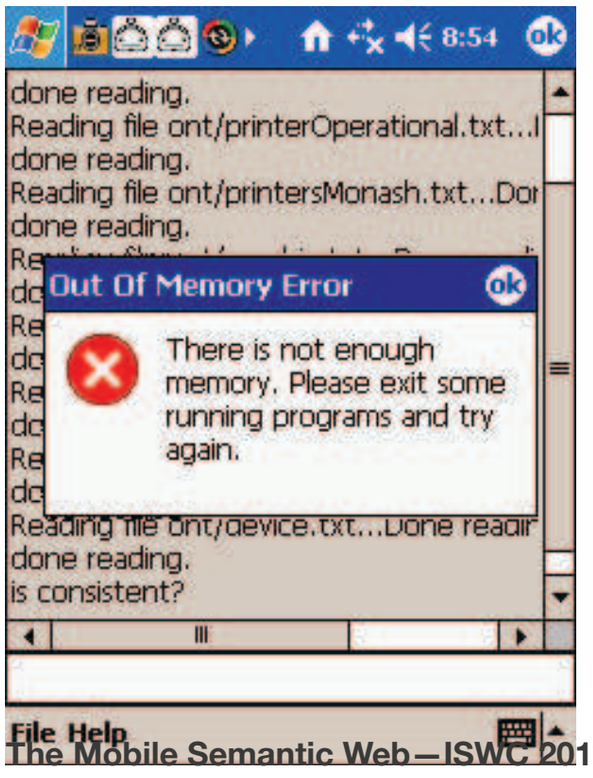
- Hence, **eliminate rule applications** to improve efficiency
  - Completeness trade-off
- Introducing **mTableaux** (Steller & Krishnaswamy (2008, 2009))

# *mTableaux* — Optimising Tableaux for Mobile Semantic Reasoning

# mTableaux – Basic Premises

- Mobile inference is important
- Mobile inference is hard
    - Computationally complex
    - Mobile devices have limited resources
    - Humans have limited attention span




# mTableaux – At A Glance

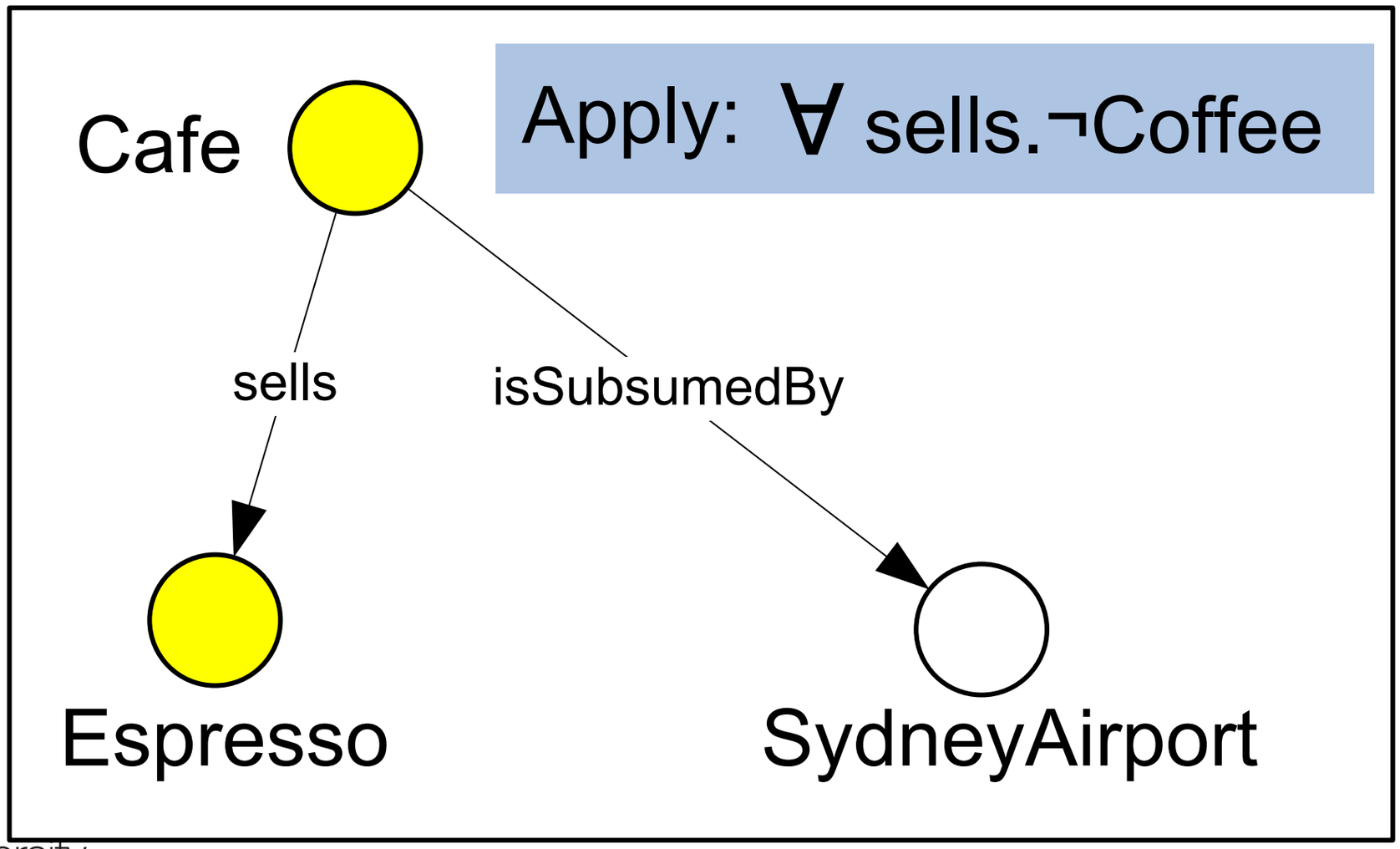
- A framework for light-weight mobile DL inference
  - In a service matching setting
- Focus: class membership checking for a single individual $C(a) \in \mathcal{A}$
  - Matching a request against a service description
- New optimisation & caching strategies
  - Selective application of transformation rules (**ST**)
  - Selective application of the disjunction rule (**SD**)
  - Caching based rule prioritisation (**CS**)
- Adaptive Inference Strategies
  - Leveraging priority order of matching in reasoning
  - Persistence to support incremental and partial reasoning
  - Resource-adaptation as a control mechanism
  - A *degree of match* metric

# mTableaux – At A Glance

- Caveat - realisation/consistecny checking not performed
  - Ontology assumed to be consistent
  - Hence no completeness guarantee
    - False negatives possible, false positives not

# Selective Transformation Rule Application (ST)

- Tableaux expansion rules are applied to only a subset of individuals, which relate to the service description being checked



# Selective Application of Expansion Rules

- Only apply transformation rules to a subset $\mathcal{ST}$ of all individuals in ABox $\mathcal{A}$
  - $\mathcal{ST}$ ***relevant*** to the inference task
  - Not applied to any other individual
- Populating $\mathcal{ST}$
  - Starting with the request individual
  - Adding other individuals iteratively

$$\mathcal{ST} = \mathcal{ST} \cup \bigcup_{x_i \in \mathcal{ST}} x_y, \text{ where}$$

$$x_y \notin \mathcal{ST} \text{ and } \forall R_r.C \in \mathcal{L}(x_i) \text{ and}$$

$$(R_r \in \mathcal{L}(\langle x_i, x_y \rangle) \text{ or } R_p \in \mathcal{L}(\langle x_i, x_y \rangle) \text{ where } R_p \sqsubseteq R_r)$$

$$\mathcal{A} \stackrel{?}{\models} (\exists sells.Internet)(netCafe)$$

$$\mathcal{A} \cup \{(\forall sells.\neg Internet)(netCafe)\} \text{ consistent?} \quad \mathcal{ST} = \{netCafe\}$$

$$sells \in \mathcal{L}(\langle netCafe, inet \rangle)$$

$$\mathcal{A} \cup \{(\neg Internet)(inet)\} \qquad \mathcal{ST} \cup \{inet\}$$

# Selective Application of Expansion Rules

- ST aims at generating clashes for the assertion
  - Only need to evaluate the **request** individual
  - And any others **changed** by transformation rules on the request
- ***Universal quantifications*** only, no other descriptions contribute to $\mathcal{ST}$

| | |
|---|---|
| $(\forall R.C)(x_i), \langle x_i, y \rangle \in R$ | Type of an existing individual ***changes*** |
| $(C_1 \sqcap C_2)(x_i)$ | $x_i$ ***already*** in $\mathcal{ST}$ |
| $(C_1 \sqcup C_2)(x_i)$ | $x_i$ ***already*** in $\mathcal{ST}$ |
| $(\exists R.C)(x_i)$ | ***New consistent*** individual generated |
| $(\geq n\ R)(x_i)$ | ***New*** individuals created without type |
| $(\leq n\ R)(x_i)$ | Existing individuals merged, clash ***unlikely*** |

# Updated Expansion Rules in ST

$\sqcap$-rule

$$(C_1 \sqcap C_2)(a) \in \mathcal{A}$$
$$C_1(a) \notin \mathcal{A} \wedge C_2(a) \notin \mathcal{A}$$
$$\underline{a \in \mathcal{ST}}$$

$$\mathcal{A}' := \mathcal{A} \cup \{C_1(a), C_2(a)\}$$

$\sqcup$-rule

$$(C_1 \sqcup C_2)(a) \in \mathcal{A}$$
$$C_1(a) \notin \mathcal{A} \wedge C_2(a) \notin \mathcal{A}$$
$$\underline{a \in \mathcal{ST}}$$

$$\mathcal{A}' := \mathcal{A} \cup \{C_1(a)\}, \mathcal{A}'' := \mathcal{A} \cup \{C_2(a)\}$$

$\exists$-rule

$$(\exists\, r.C)(a) \in \mathcal{A}$$
$$\nexists\, z \in \mathcal{A} \bullet C(z) \in \mathcal{A} \wedge r(a, z) \in \mathcal{A}$$
$$\underline{a \in \mathcal{ST}}$$

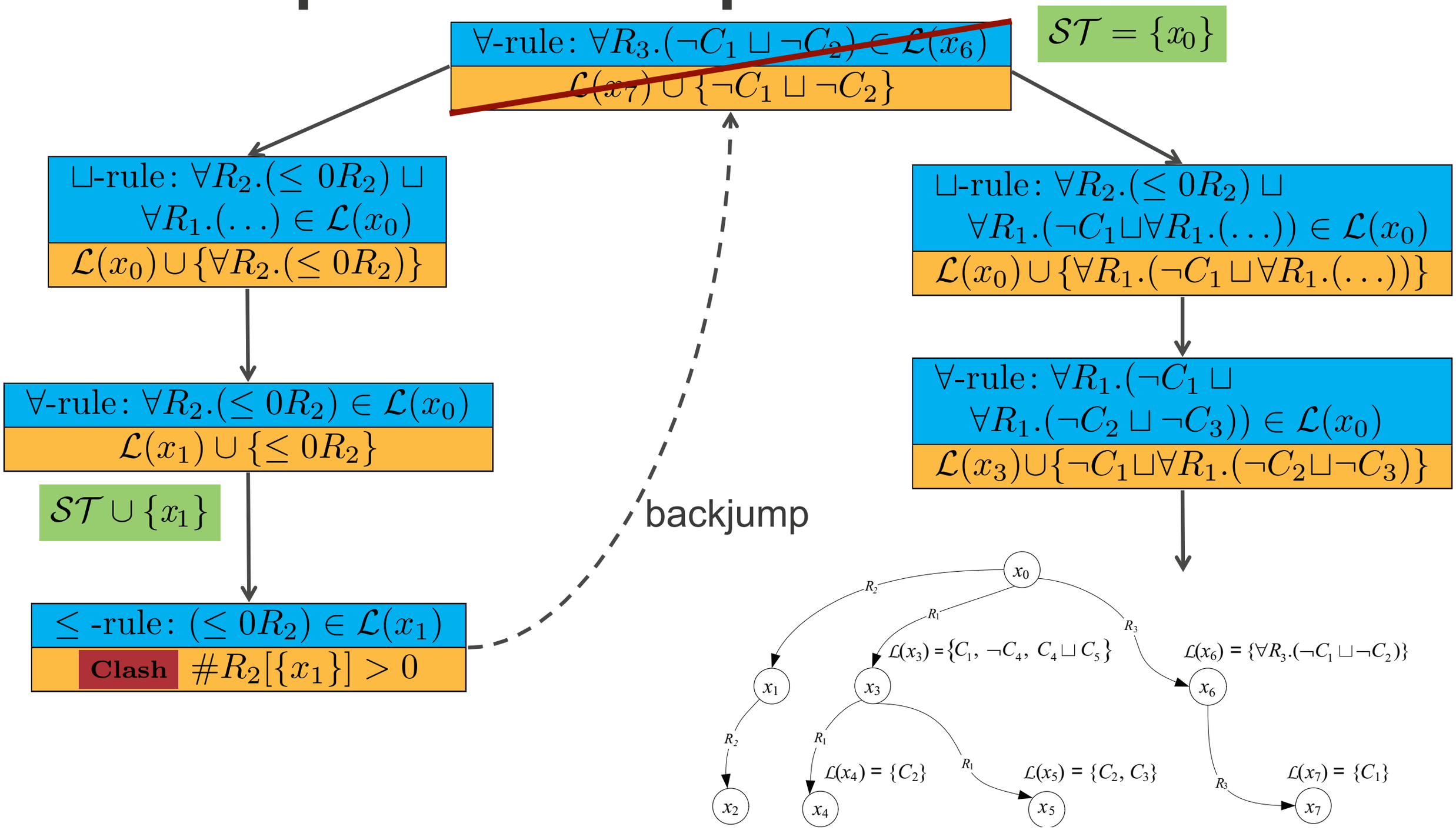
$$\mathcal{A}' := \mathcal{A} \cup \{C(b), r(a, b)\}$$

$\forall$-rule

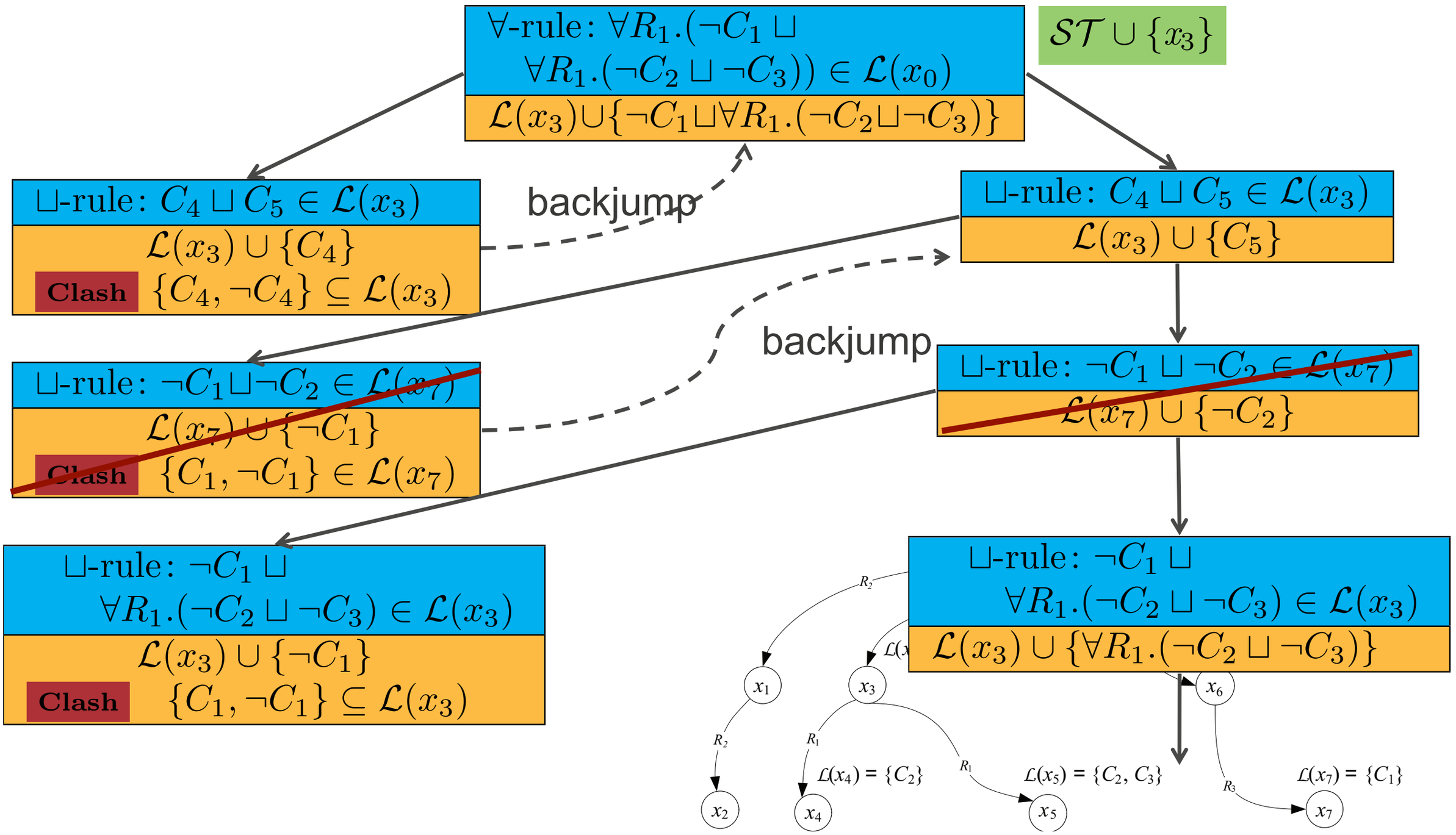
$$(\forall\, r.C)(a) \in \mathcal{A} \wedge r(a, b) \in \mathcal{A}$$
$$C(b) \notin \mathcal{A}$$
$$\underline{a \in \mathcal{ST}}$$

$$\mathcal{A}' := \mathcal{A} \cup \{C(b)\}$$

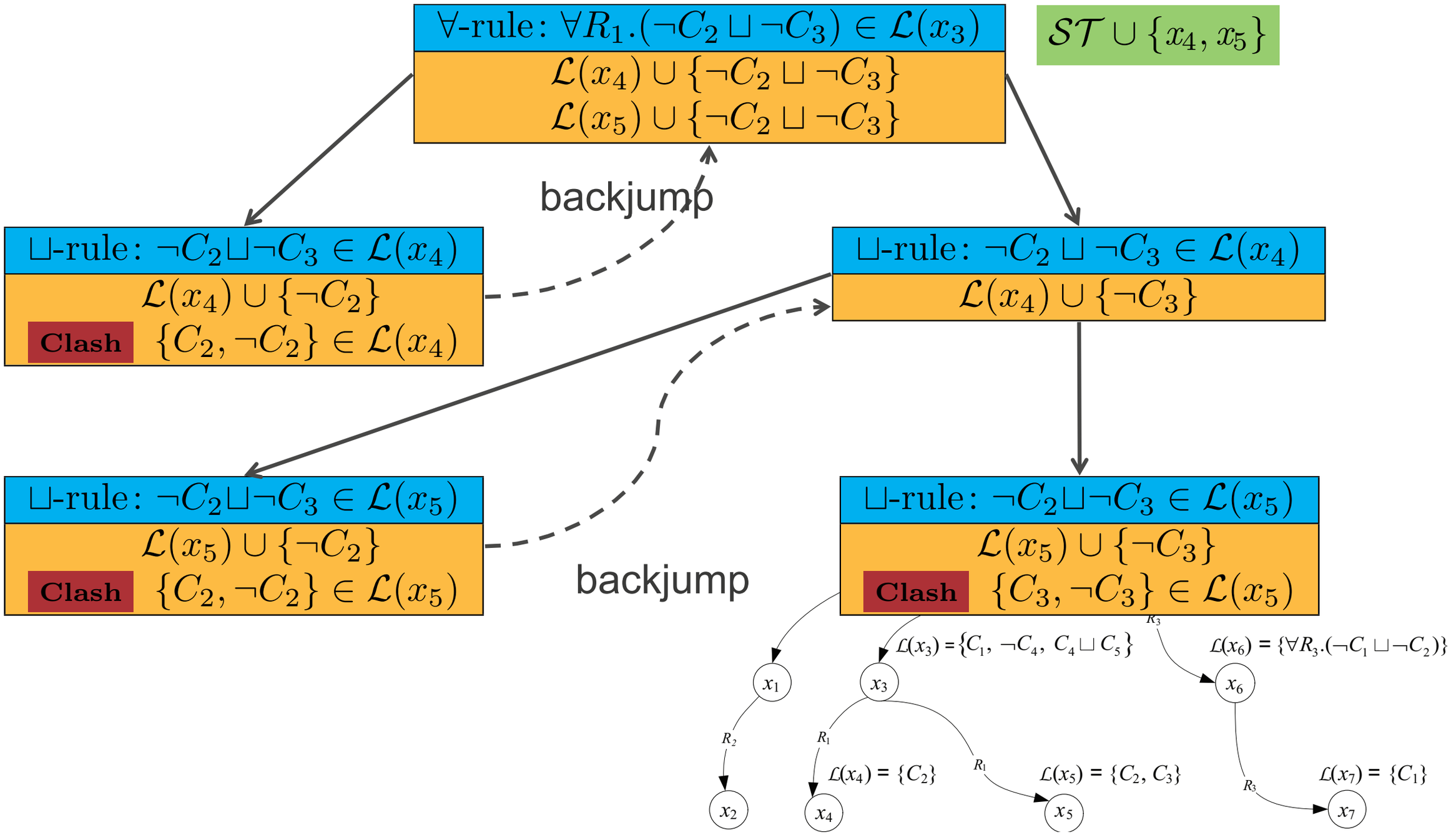
# ST – Updated Example



# ST – Updated Example



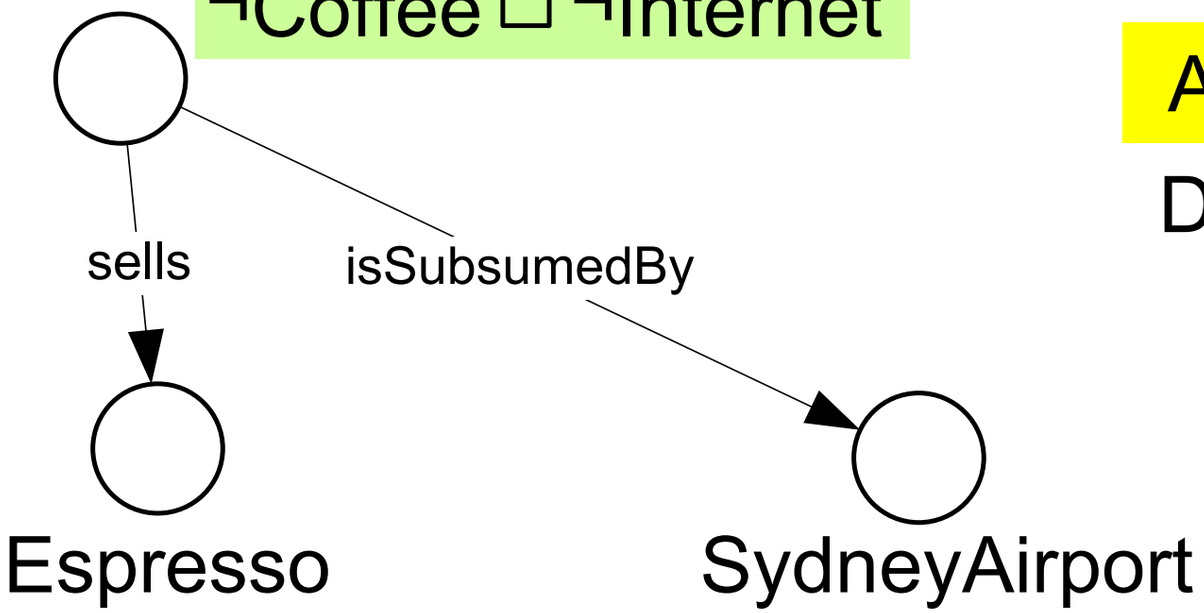
# ST – Updated Example



# Selective Disjunction Rule Application (SD)

- Tableaux disjunction expansion rule is only applied to disjunctions which contain class concepts which relate to the user request

User Request:
¬Coffee ⊔ ¬Internet




Apply: (Coffee ⊔ Tea)

Don’t Apply: (Pizza ⊔ VideoGame)

# Selective Disjunction Rule Application

$\sqcup$-rule

$$(C_1 \sqcup C_2)(a) \in \mathcal{A}$$

$$C_1(a) \notin \mathcal{A} \wedge C_2(a) \notin \mathcal{A}$$

$$\underline{a \in \mathcal{ST}}$$

---

$$\mathcal{A}' := \mathcal{A} \cup \{C_1(a), C_2(a)\}, \mathcal{A}'' := \mathcal{A} \cup \{C_2(a)\}$$

- The Disjunction rule is **evil**
  - It generates new ABoxes, all of which need to be explored
  - Hence increases search space
- SD – only apply the disjunction rule to a subset $\mathcal{SD}$ of class descriptions
  - Those containing a description in the *user request*
  - Hence reducing search space
- *E.g.,* looking for Internet café? $\exists\, sells.Internet \sqcap \exists\, sells.Coffee$
  - Apply $\sqcup$-rule to $Tea \sqcup Coffee$, not to $Pizza \sqcup VideoGame$

# Selective Disjunction Rule Application

- Populating $\mathcal{SD}$ recursively

**Algorithm 1** $GenSD(D, \mathcal{SD})$

**Inputs:** ClassConcept $C_i$
**Output:** Set $\mathcal{SD}$

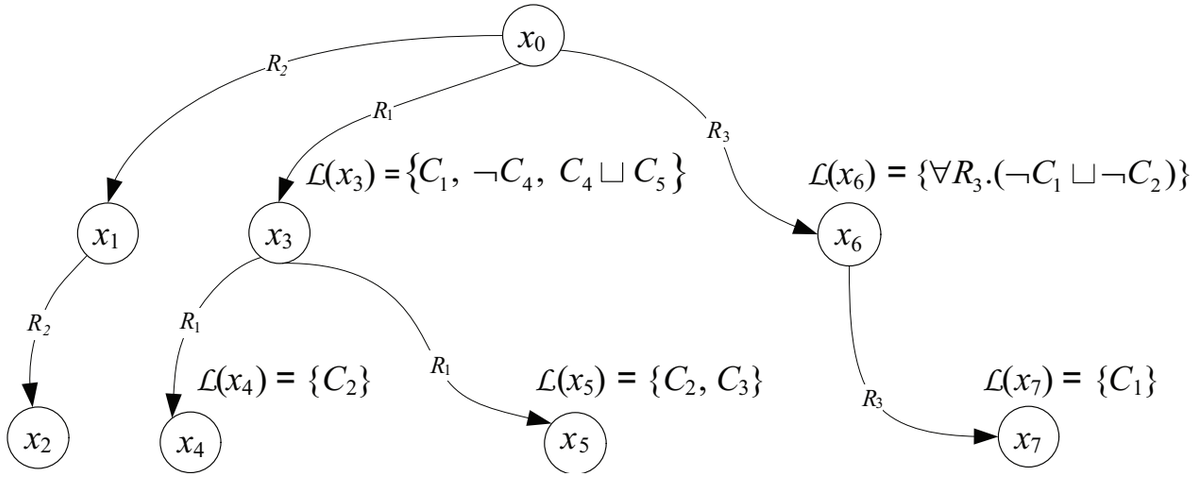
1: **let** $\mathcal{SD} \leftarrow \emptyset$
2: **if** $C_i = \neg C_j$ **then**
3: $\quad \mathcal{SD} \leftarrow \{C_j\}$
4: **else**
5: $\quad \mathcal{SD} \leftarrow \{C_i\}$
6: **end if**
7: **if** $C_i = \forall R.C_j$ **then**
8: $\quad \mathcal{SD} \leftarrow \mathcal{SD} \cup GenSD(C_j)$
9: **else if** $C_i = \exists R.C_j$ **then**
10: $\quad \mathcal{SD} \leftarrow \{C_i\}$
10: $\quad \mathcal{SD} \leftarrow \{C_i\}$
11: **else if** $C_i = C_1 \sqcap \ldots \sqcap C_m$ **then**
12: $\quad$ **for all** $C_j \in \{C_1, \ldots, C_m\}$ **do**
13: $\quad\quad \mathcal{SD} \leftarrow \mathcal{SD} \cup GenSD(Cj)$
14: $\quad$ **end for**
15: **else if** $C_i = C_1 \sqcup \ldots \sqcup C_m$ **then**
16: $\quad$ **for all** $C_j \in \{C_1, \ldots, C_m\}$ **do**
17: $\quad\quad \mathcal{SD} \leftarrow \mathcal{SD} \cup GenSD(C_j)$
18: $\quad$ **end for**
19: **end if**
20: **for all** $C_j \sqsubseteq C_i$ **do**
21: $\quad \mathcal{SD} \leftarrow \mathcal{SD} \cup \{C_j\}$
22: **end for**
23: **return** $\mathcal{SD}$

# Selective Disjunction Rule Application

- Populating $\mathcal{SD}$ - an example

$$\mathcal{A} \cup ((\forall\, R_2.(\leq 0 R_2)) \sqcup (\forall\, R_1.(\neg\, C_1 \sqcup \forall\, R_1.(\neg\, C_2 \sqcup \neg\, C_3)))(x_0) \text{ is closed?}$$

$$\mathcal{SD} = \begin{array}{l} (\forall\, R_2.(\leq 0 R_2)) \sqcup (\forall\, R_1.(\neg\, C_1 \sqcup \forall\, R_1.(\neg\, C_2 \sqcup \neg\, C_3))) \\ \forall\, R_2.(\leq 0 R_2), \forall\, R_1.(\neg\, C_1 \sqcup \forall\, R_1.(\neg\, C_2 \sqcup \neg\, C_3)) \\ \leq 0 R_2, \neg\, C_1 \sqcup \forall\, R_1.(\neg\, C_2 \sqcup \neg\, C_3) \\ \forall\, R_1.(\neg\, C_2 \sqcup \neg\, C_3) \\ \neg\, C_2 \sqcup \neg\, C_3 \\ C_1, C2, C_3 \end{array}$$



# Selective Disjunction Rule Application

- The SD optimisation

**Algorithm 1** ApplyDisj($D, \mathcal{SD}$)

**Inputs:** ClassConcept $C_i$, set $\mathcal{SD}$
**Output:** Boolean $applyDisj$

1: **for all** $C_i$ from $D$ where $D = C_1 \sqcup \ldots \sqcup C_n, 1 \leq i \leq n$ **do**
2: **if** $C_i \in \mathcal{SD}$ **then**
3: **return** `true`
4: **else if** $C_j \in \mathcal{SD}$, for any $C_j \sqsubseteq C_i$ **then**
5: **return** `true`
6: **end if**
7: **end for**
8: **return** `false`

# Selective Disjunction Rule Application

- The new disjunction rule

$\sqcup$-rule

$$(C_1 \sqcup C_2)(a) \in \mathcal{A}$$

$$C_1(a) \notin \mathcal{A} \wedge C_2(a) \notin \mathcal{A}$$

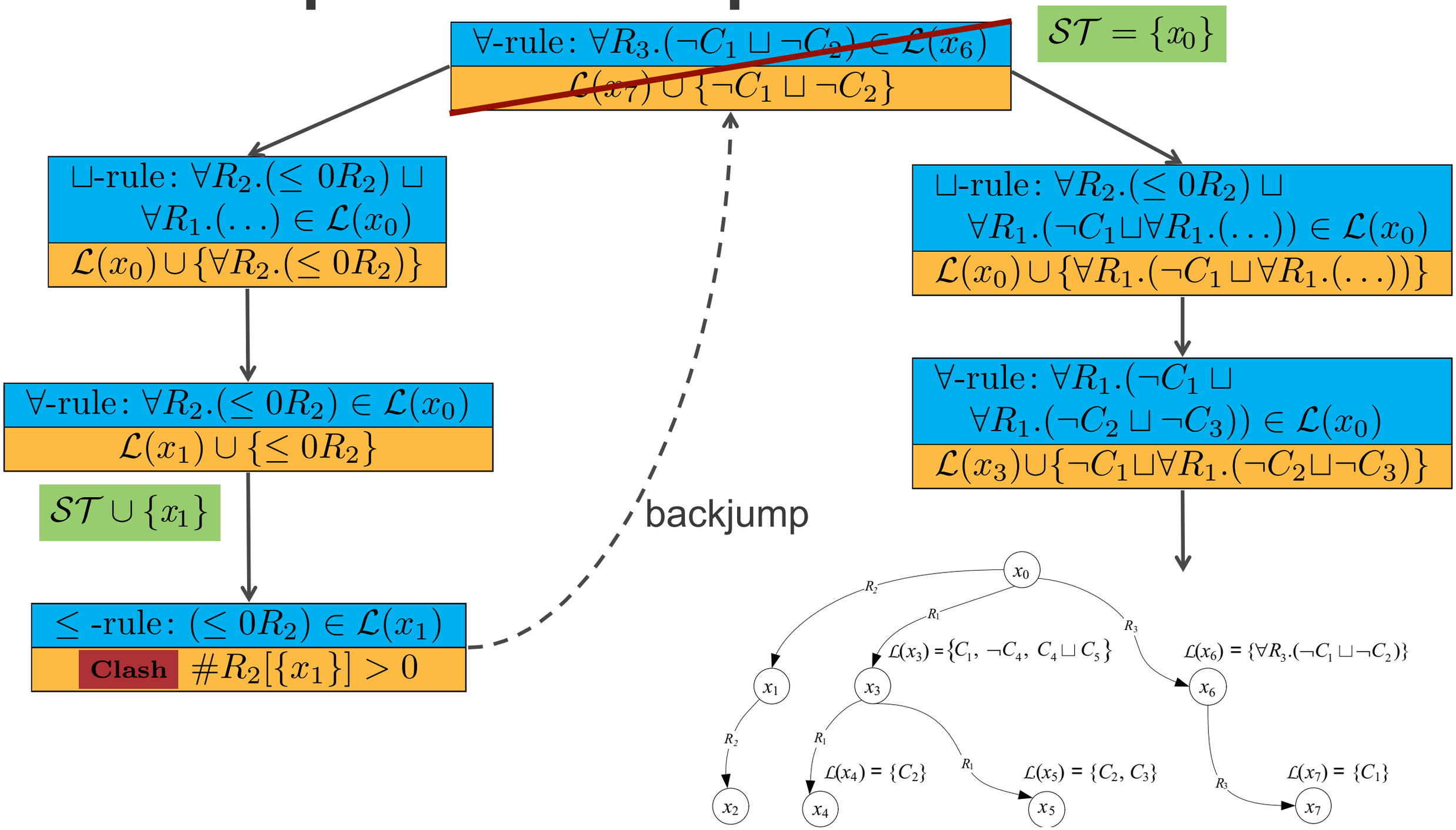
$$a \in \mathcal{ST}$$

$$ApplyDisj((C_1 \sqcup \ldots \sqcup C_n), \mathcal{SD}) = \texttt{true}$$

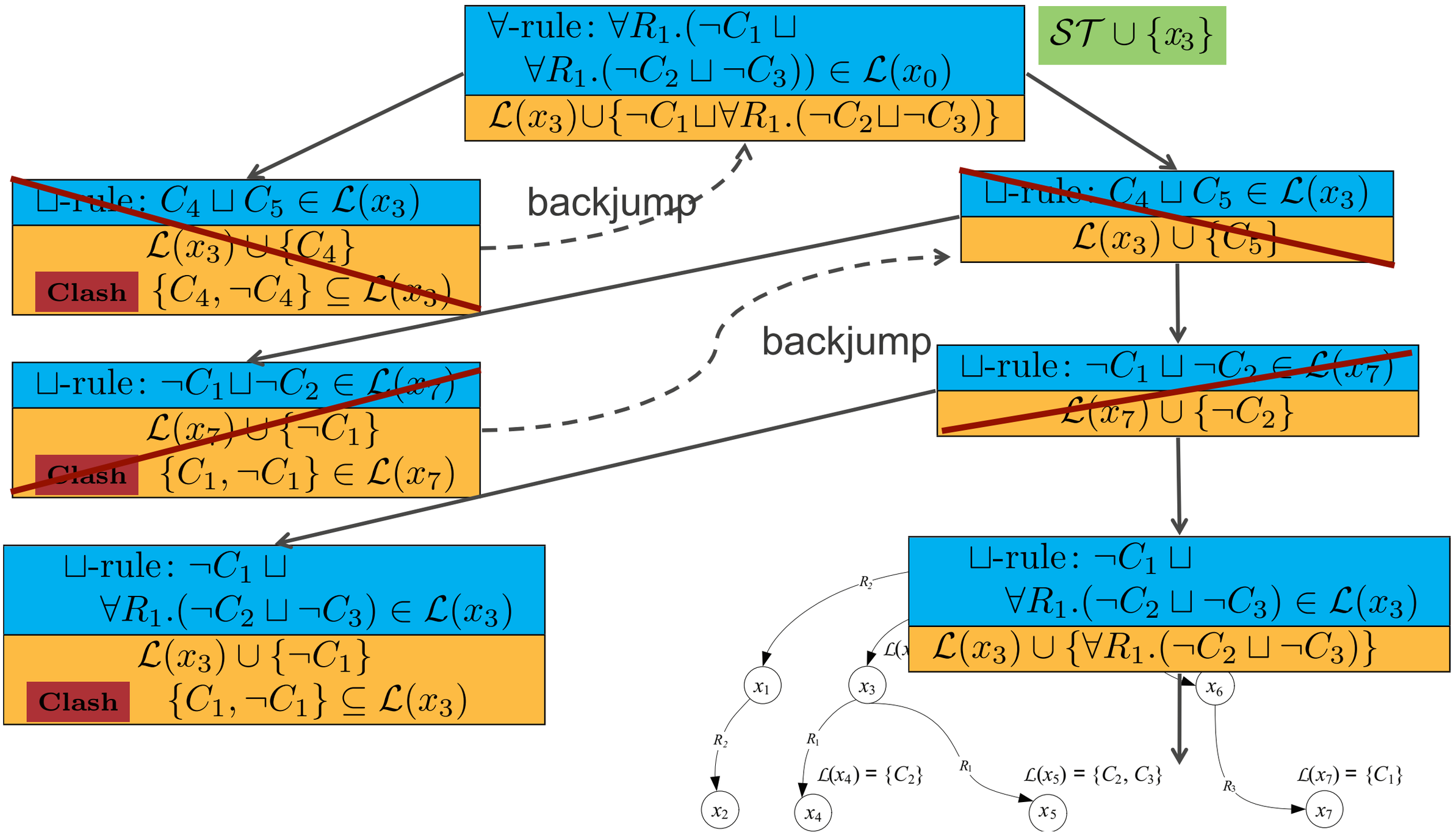
---

$$\mathcal{A}' := \mathcal{A} \cup \{C_1(a)\}, \mathcal{A}'' := \mathcal{A} \cup \{C_2(a)\}$$

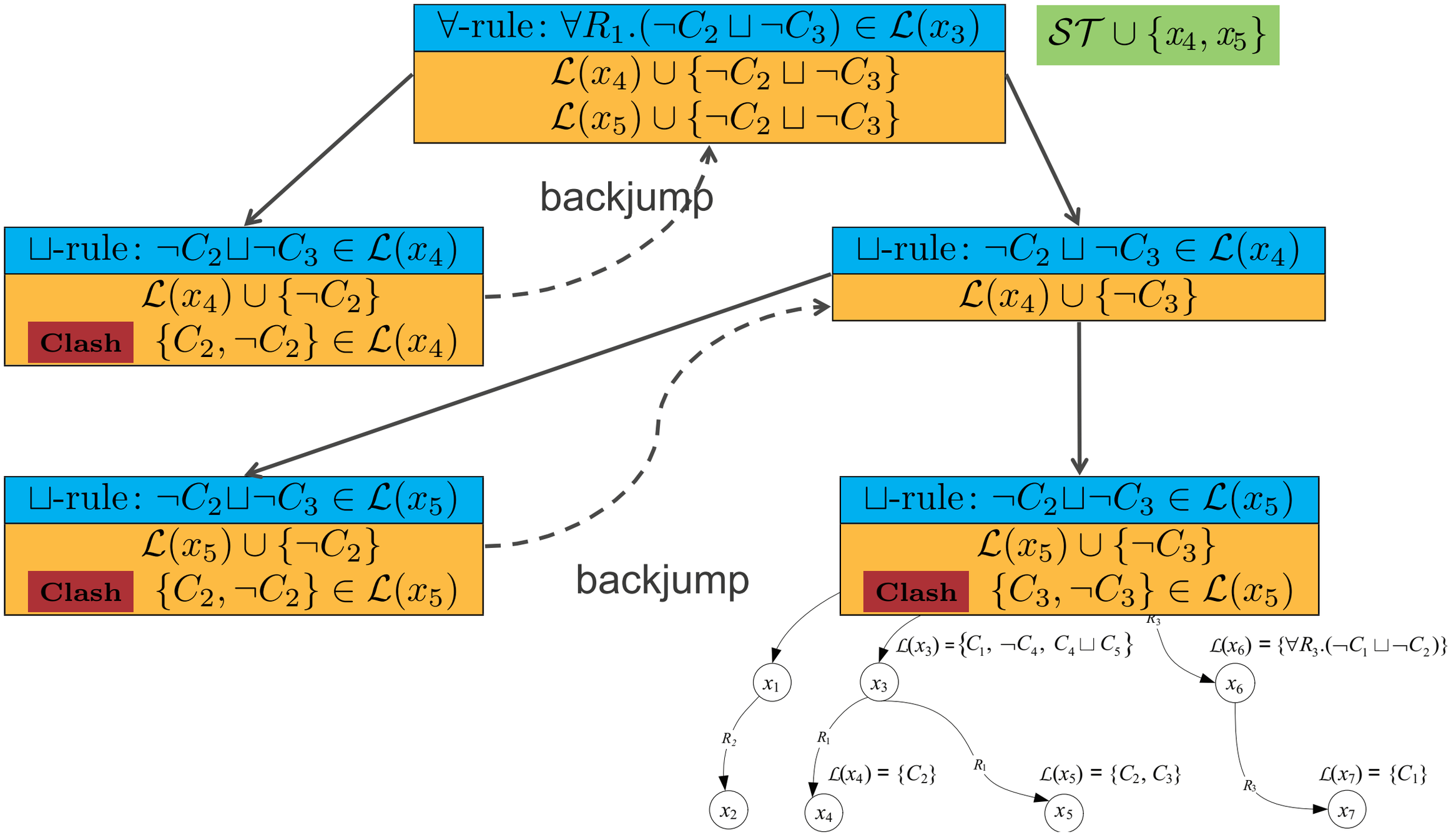
# SD – Updated Example



# SD – Updated Example



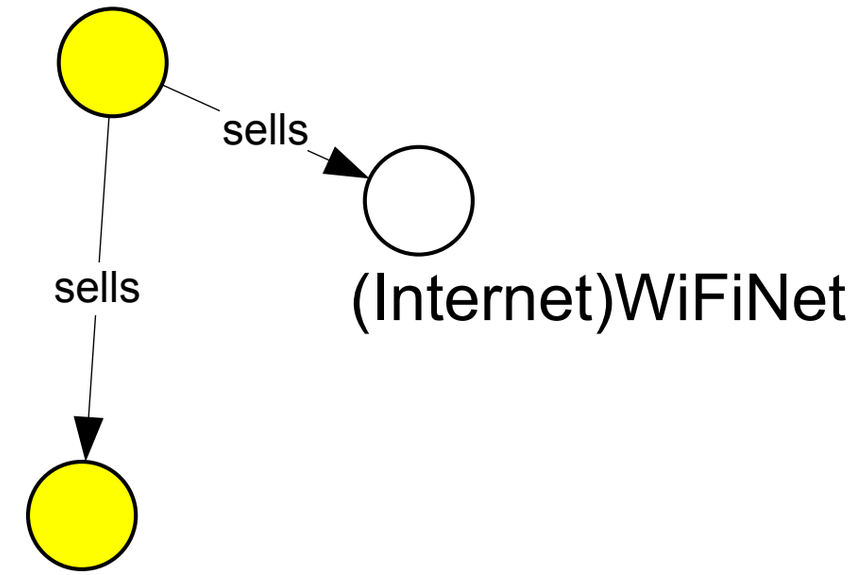
# SD – Updated Example



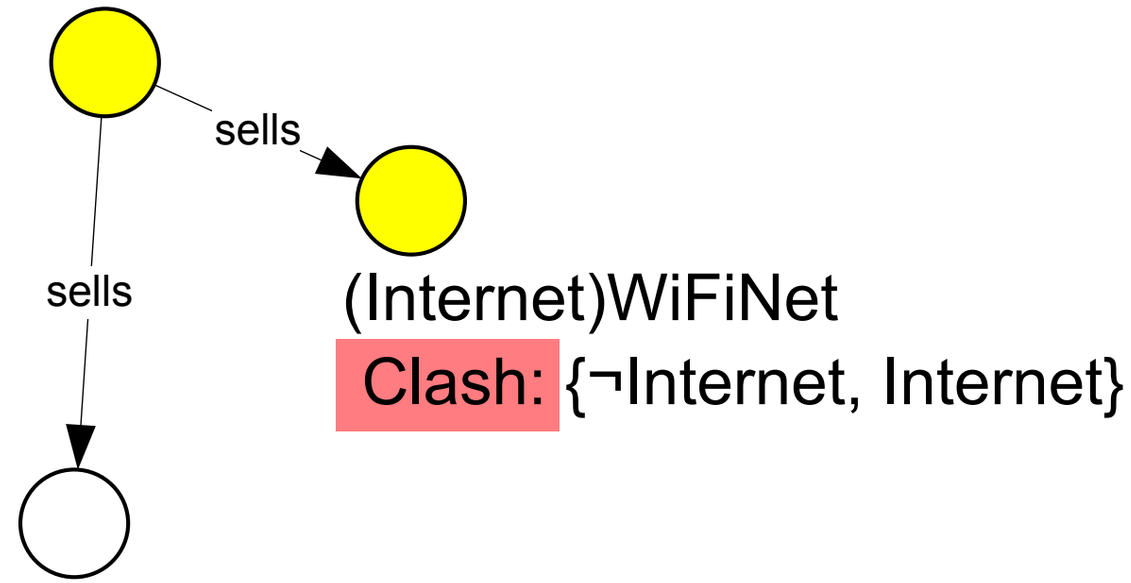
# Caching Strategy (CS)

- Tableaux transformation rules which contribute to a clash (i.e. a positive match) are cached

User Request A:
∀ sells.¬Coffee




User Request B:
∀ sells.(¬Coffee ⊔ ¬Internet)



# Caching Strategy

- There may still be transformation rules that don't contribute to the task

| $\forall$-rule: $\forall R_1.(\neg C_2 \sqcup \neg C_3) \in \mathcal{L}(x_3)$ |
| --- |
| $\mathcal{L}(x_4) \cup \{\neg C_2 \sqcup \neg C_3\}$ |
| $\mathcal{L}(x_5) \cup \{\neg C_2 \sqcup \neg C_3\}$ |

  - Only one of the above needs to generate a clash
    - But both need to be evaluated
    - Remembering previous clashes may help
  - Caching Strategy (CS) – avoid re-evaluation of previous assertions
    - Across inference tasks
    - In the service matching setting
    - Time sensitive
    - Sound, incomplete

# Caching Strategy

- Caching already employed by existing OWL reasoners
  - Main memory only
- In a mobile setting
  - ***Similar*** requests may be made ***successively***
  - ***Limited*** main memory
  - Substantial ***fast*** secondary storage (flash-based)
  - Hence, ***persistent*** caching ***across*** tasks may improve performance

| **Cache entry** | | **Timestamp** | |
|---|---|---|---|
| | | Valid | Expired |
| **Matching** | Full | Immediate clash | Evaluated early |
| | Partial | Evaluated early | Evaluated early |

# Cache Structure

- Cache $CSCache = \{\zeta_1, \ldots, \zeta_n\}$, where

  $\zeta = \langle C_i(x_r), C_j(x_p), t, s \rangle$, such that

  $C_j(x_p)$ is the assertion which depends on the assertion $C_i(x_r)$,

  $t$ is the timestamp of $\zeta$ addition, and

  $s$ is `true` if all possibilities for $C_i(x_r)$ clashed or `false` otherwise.

- *E.g.,* looking for Internet? $\neg\ InternetRequest \equiv \forall\, hasComm.(\neg\ WiFi) \sqcup \neg\ Internet$
  - Assuming that $\neg\ WiFi$ clashes at time $t$

$$CSCache \leftarrow \{\langle \neg\ WiFi, \texttt{null}, t, \texttt{true} \rangle\}$$

depends

$$CSCache \leftarrow CSCache \cup \{\langle \forall\, hasComm.(\neg\ WiFi), \neg\ WiFi, t, \texttt{true} \rangle\}$$

depends

$$CSCache \leftarrow CSCache \cup \{\langle \neg\ InternetRequest, \forall\, hasComm.(\neg\ WiFi), \texttt{null}, t, \texttt{false} \rangle\}$$

# Cache Storage

**Algorithm 1**
$ClashDetected(C_j(x_p), status)$

**Inputs:** Assertion $C_j(x_p)$, Boolean $status$

1: **if** $status = \texttt{true}$ **then**
2: $\quad AddCacheEntry(C_j(x_p), \texttt{null}, \texttt{true})$
3: $\quad status \leftarrow \texttt{false}$
4: **end if**
5: **let** $C_i(x_r) \leftarrow dependsOn(C_j(x_p))$
6: **if** $C_i(x_r) \neq \texttt{null}$ **then**
7: $\quad$ **let** $fullClash \leftarrow \texttt{false}$
8: $\quad$ **if** $(C_i(x_r) = (C_1 \sqcup \ldots \sqcup C_n)(x_r)) \wedge$
$(s = \texttt{true},$ for all $\langle C_k(x_r), \_, s, \_\rangle \in$
$CSCache,$ where $1 \leq k \leq n)$ **then**
9: $\quad\quad fullClash \leftarrow \texttt{true}$
10: $\quad$ **end if**
11: $\quad AddCacheEntry(C_i(x_r), C_j(x_p), fullClash)$
12: $\quad ClashDetected(C_i(x_r), status)$
13: **end if**

# Adaptive Inference Strategy

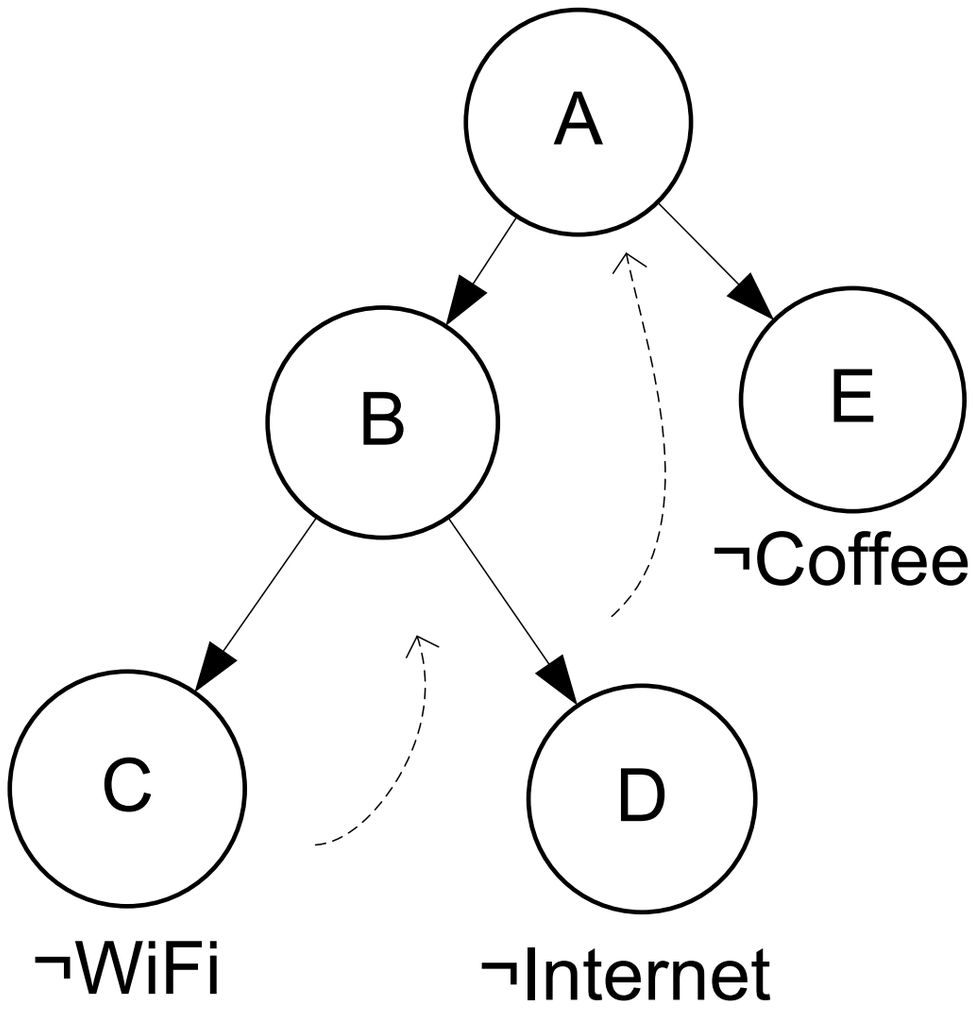
- Typically Current Reasoners:
  - Depth-first / arbitrary ordering of user requirements
  - Complete entire inference process before a result is provided: “all or nothing” principle
  - Provide a match / no-match result

- Adaptive Inference Strategy:
  - Leveraging priority order of matching in reasoning
  - Persistence to support incremental and partial reasoning
  - Resource-adaptation as a control mechanism
  - A *degree of match* metric

# Standard vs Adaptive Inference

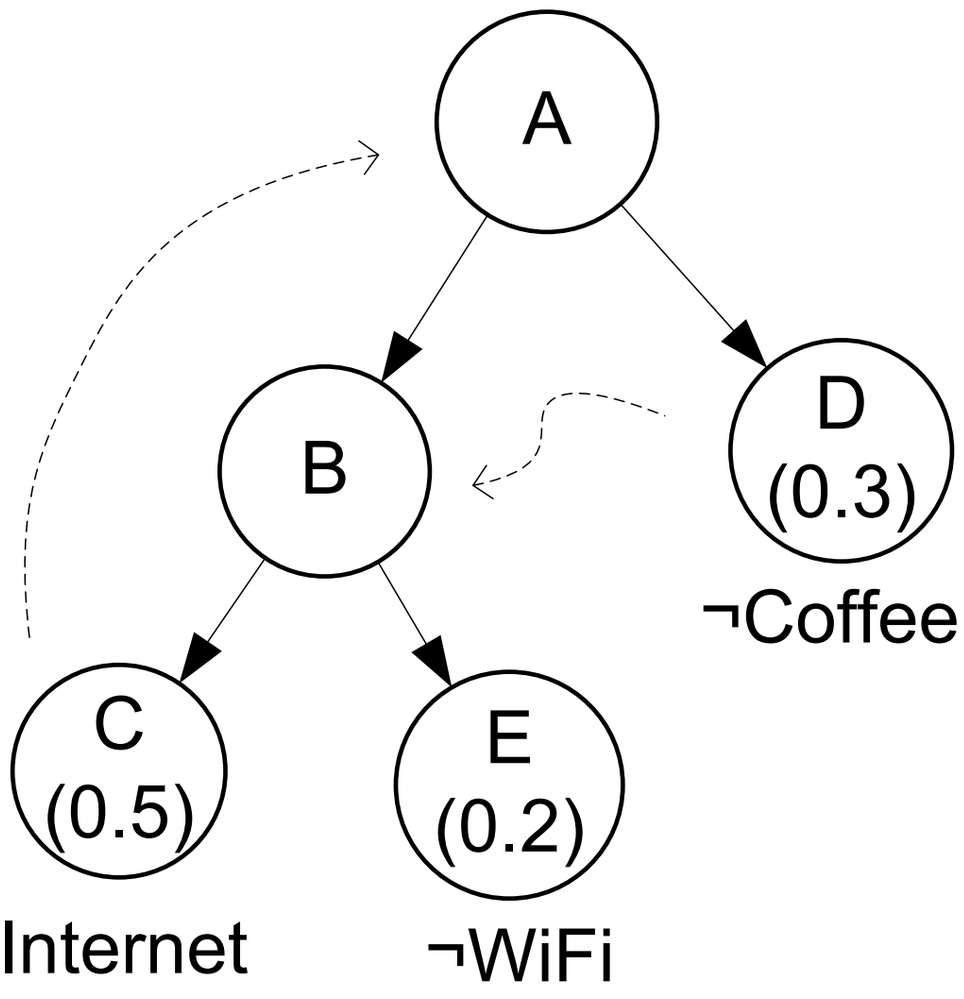
User Request: ((¬WiFi ⊔ ¬Internet) ⊔ ¬Coffee)

Standard Tableaux:




Adaptive Inference:

Order: ¬Internet, ¬Coffee, ¬WiFi



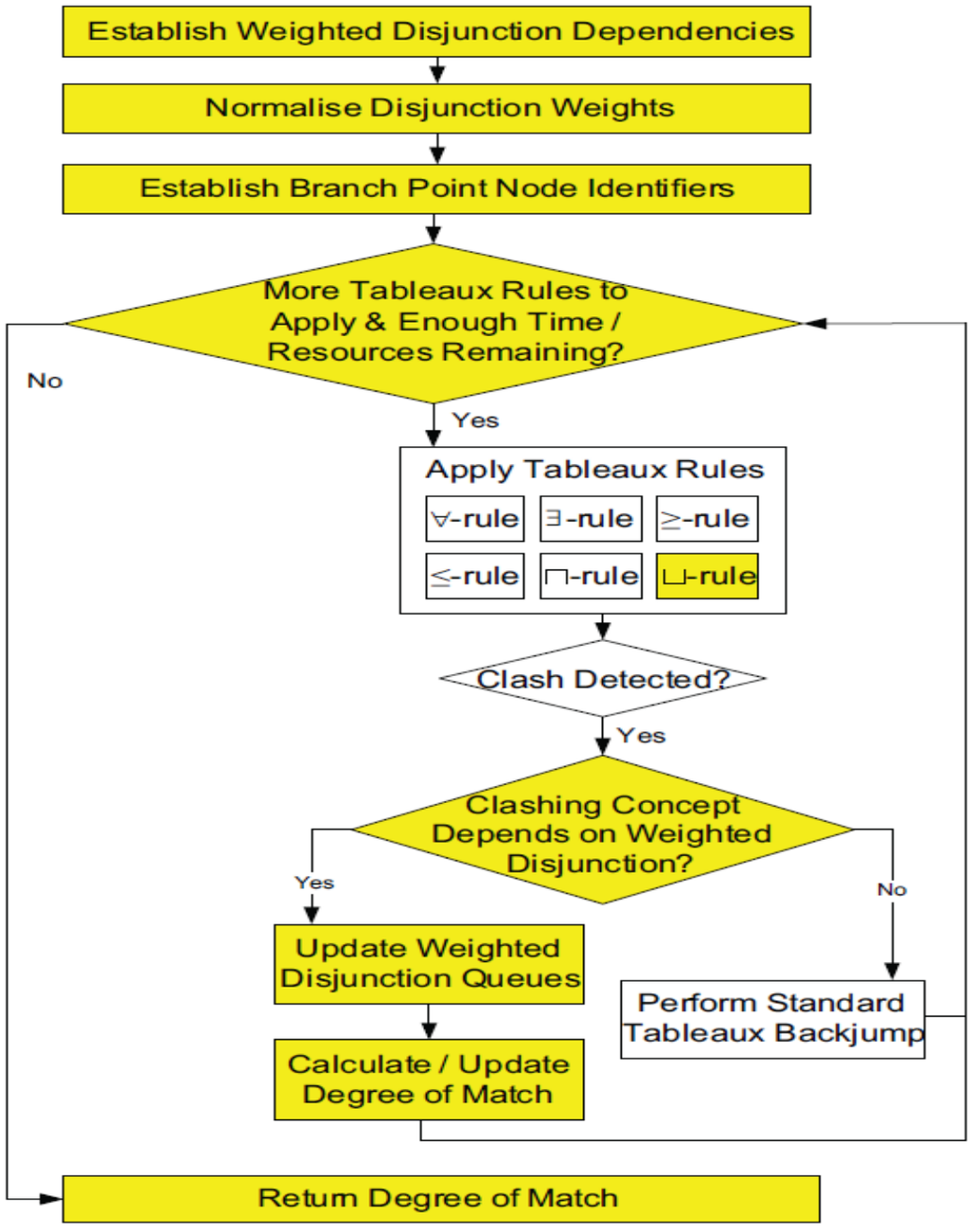
# Adaptive Inference Strategy - Considerations

- Priority Expansion of Disjunction Elements
  - Queuing and Expansion Ordering

- Simultaneously Open / Unfinished Branches
  - Branch Identifiers
  - State Management

- Degree of Match Computation

# Adaptive Inference Process – A Systems View



# Adaptive Inference Strategy

## ➢ Priority Expansion of Disjunction Elements

- ⊔-rule is applied to disjunctive assertions -> Tableaux tree *G* expansion
- Each disjunct element of the disjunction represents an alternative expansion possibility to check
- An inference is only proven if every tree expansion which depends on the request *C* in the inference check, generates a clash
- Each conjunction in *C* for the inference check *C*(*x*), will after negation, result in the creation of a new branch point node in *G*

# Adaptive Inference Strategy

## ➢ Priority Expansion of Disjunction Elements

- Each conjunct element that is a member of any of these conjunctions will result in a new branch edge in *G*
- We  associating weights of importance with each conjunct element in the request
- Matching must be in conjunct priority order
- This means tree expansion order will be driven by weights rather than depth first without any specific order

# Adaptive Inference Strategy

- **Simultaneously Open / Unfinished Branches**
  - Branch Identifiers
  - State Management
- The disjunction rule has been modified as part of our adaptive inference strategy to take account of the user specified weights, associated with disjunctions, in deciding which disjunction to apply next.
- This involves the use of several queues / ordering mechanisms and selection priorities.
- If the application of Tableaux rule generates a clash then our adaptive backjump process is invoked.

# Adaptive Inference Strategy

- **Simultaneously Open / Unfinished Branches**
  - Branch Identifiers
  - State Management
- If the clashing concept depends on a weighted disjunction then certain updates to the disjunction queues occur.
- Additionally, a clash for a weighted disjunction results in the degree of match update/computation.
- For non-weight disjunctions, if a clash occurs for a non-weighted disjunction then standard Tableaux backjumping occurs.

# Adaptive Inference Strategy

## ➢ Simultaneously Open / Unfinished Branches

- When a clash occurs in the depth-first reasoning, it jumps back to an earlier branch and all branches and assertions which occurred after this branch can be discarded, because the branch is fully expanded.
- However, in our adaptive inference strategy, there may still be several unfinished branches when a backjump occurs, which the reasoner will return to finish later, if there is enough time / resources.
- Thus, branches cannot be discarded when a backjump occurs, unless the branch is fully expanded.

# Adaptive Inference Strategy - Considerations

## ➢ Degree of Match Computation

- Normalised sum of branch weights at the point of interruption or completion of reasoning
- Allows adaptation wrt to multiple constraints
  - time, resources, confidence of match

# *mTableaux* — Experimental Evaluation

# Evaluation Questions

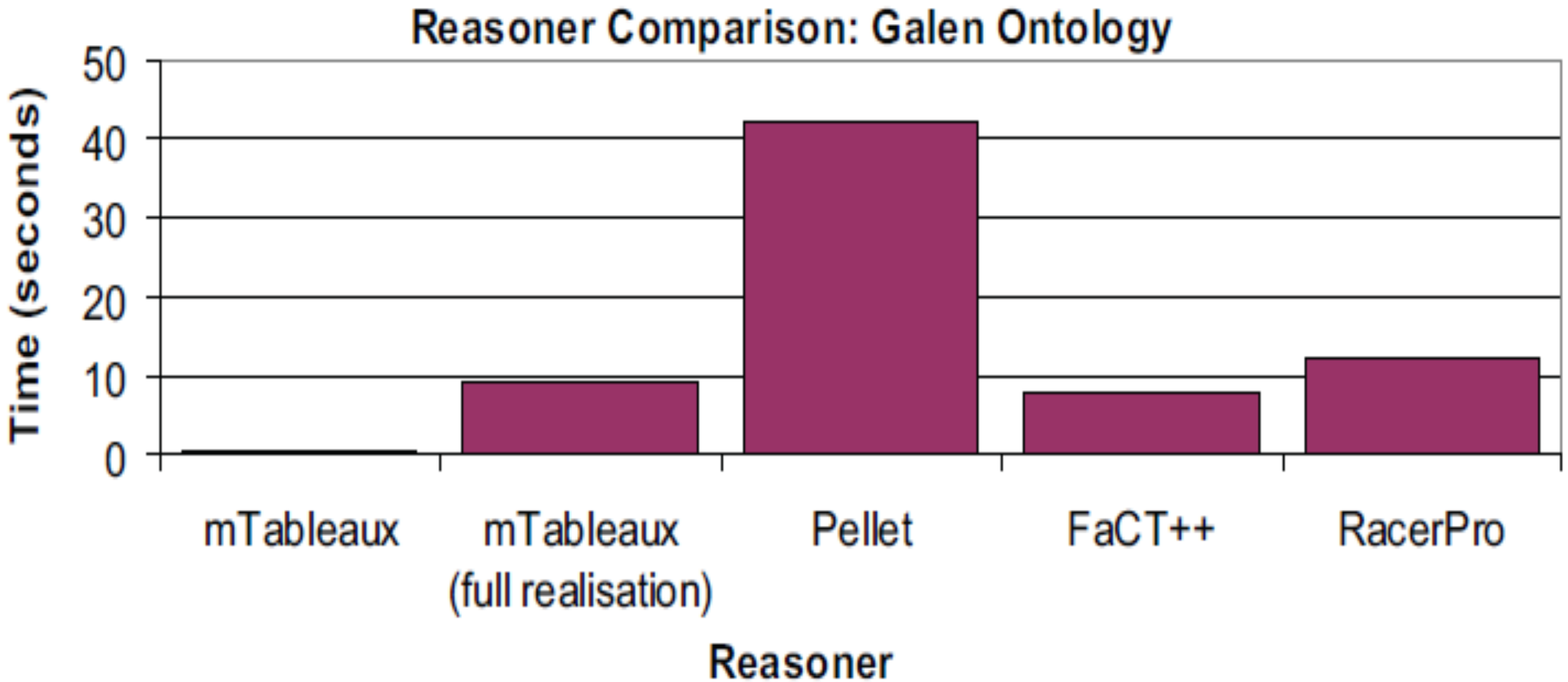
- **How does mTableaux compare with other widely available reasoners ?**
  - processing time efficiency
  - Accuracy - particularly since mTableaux optimisation strategies do not guarantee completeness,
  - what is the impact of mTableaux on accuracy?
- **2. How does mTableaux perform on a resource constrained mobile device ?**
  - in terms of performance efficiency?
- Efficiency – publicly available ontologies
- Accuracy – Case Studies – Printer Ontology and Product Ontology (developed and populated for evaluation)

# Evaluation Questions

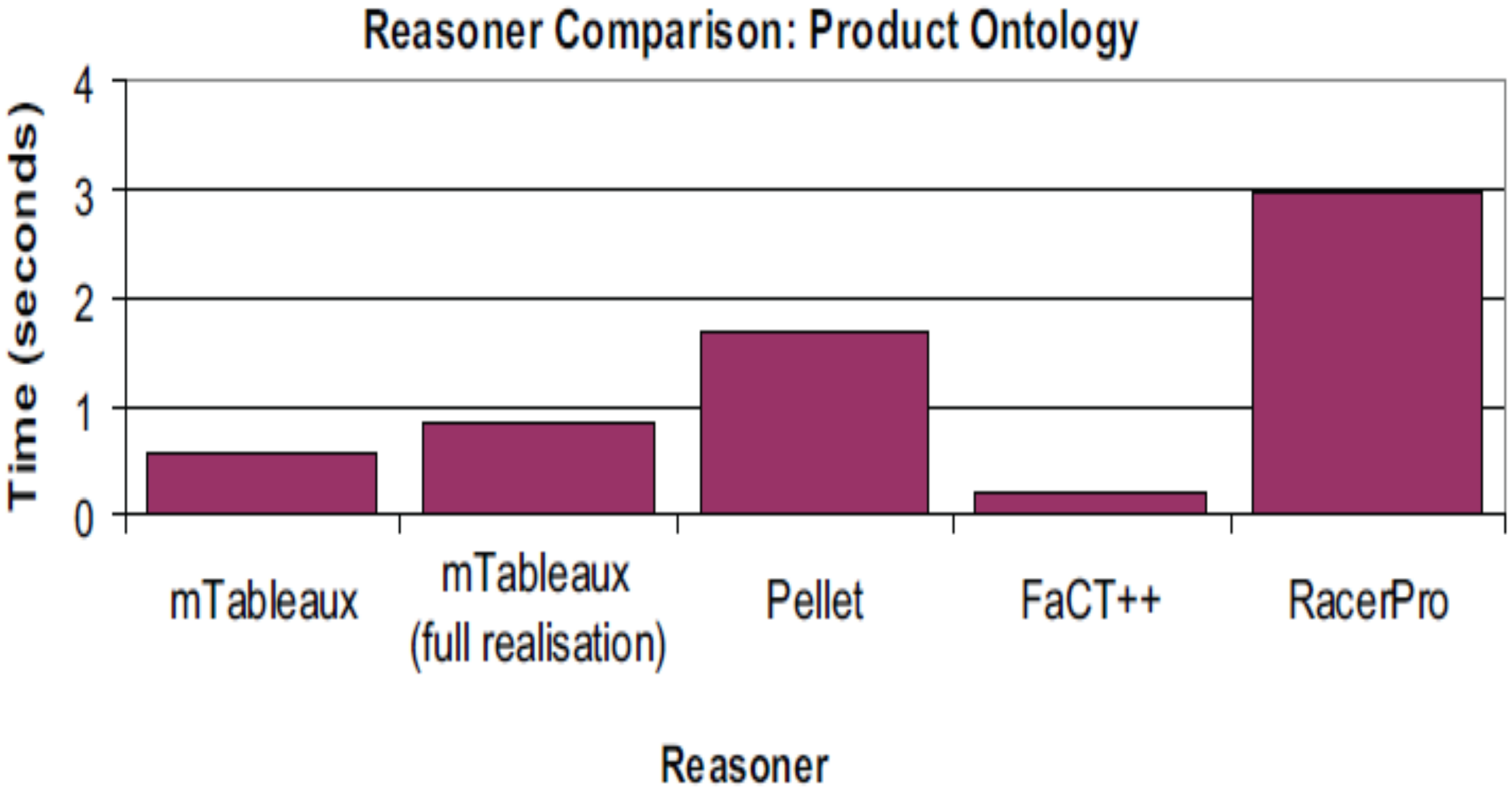
- **Compared Reasoners (in Desktop Environment – since they do not operate on mobile devices:**
  - Fact++, RacerPro, Pellet

- **Ontologies:**
  - Public:
  - Galen ontology which defines 2751 classes and 416 roles;
  - Tambis ontology which contains 188 class concept definitions and 44 role definitions;
  - Koala15 ontology which contains 23 classes and 5 roles;
  - Teams16 ontology which contains 9 classes and 3 roles.
  - Our Case Study:
    - Printer Ontology
    - Product Ontology

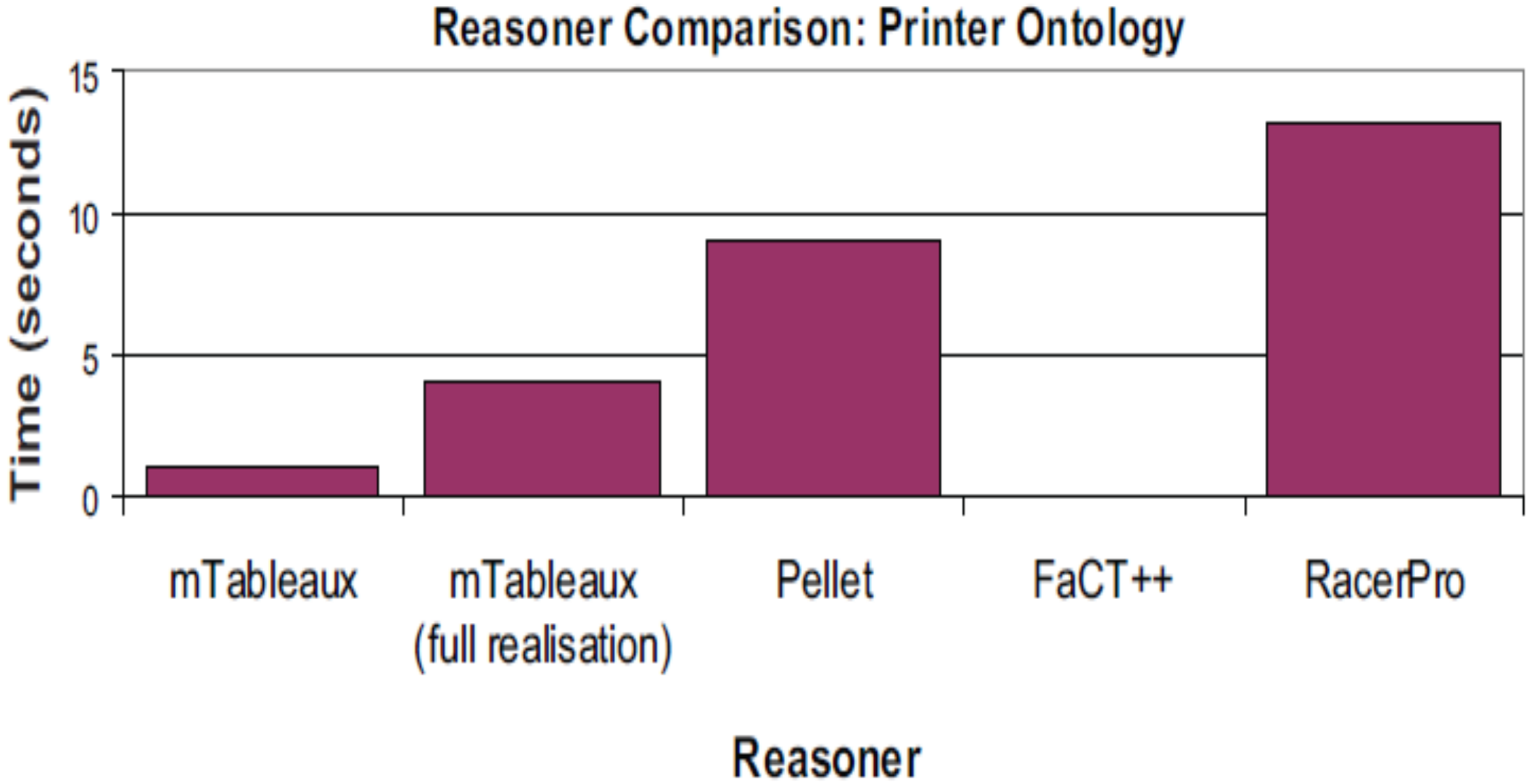
# Evaluation


Reasoner Comparison: Galen Ontology
Time (seconds)
50
40
30
20
10
0
mTableaux
mTableaux (full realisation)
Pellet
FaCT++
RacerPro
Reasoner

# Evaluation


Reasoner Comparison: Product Ontology
Time (seconds)
4
3
2
1
0
mTableaux
mTableaux (full realisation)
Pellet
FaCT++
RacerPro
Reasoner

# Evaluation


Reasoner Comparison: Printer Ontology
Time (seconds)
15
10
5
0
mTableaux
mTableaux (full realisation)
Pellet
FaCT++
RacerPro
Reasoner

# Evaluation

| Reasoner | Actual Positive | Actual Negative | Recall | Precision |
|---|---|---|---|---|
| mTableaux | 16 | 44 | $16/16 = 1.0$ | $16/16 = 1.0$ |
| Pellet | 16 | 44 | $16/16 = 1.0$ | $16/16 = 1.0$ |
| RacerPro | 16 | 44 | $16/16 = 1.0$ | $16/16 = 1.0$ |
| FaCT++ | 15 | 45 | $15/16 = 0.937$ | $15/15 = 1.0$ |

# Evaluation Questions

- mTableaux is more efficient than other open source and commercial such as the RacerPro and Pellet reasoners.
- It performs comparatively with FaCT++ when full realisation is used, and performs faster than FaCT++ when full realisation is not used
- In our tests accuracy as measured by recall and precision was not reduced by mTableaux.
- Therefore, we conclude that mTableaux does not significantly reduce accuracy in realistic data sets such as those in our evaluation.

# Evaluation – Mobile Device

- Does mTableaux enable successful completion of a matching task, such that a result can be obtained (i.e. was available memory was not exceeded)?
- Does mTableaux significantly improve performance compared to standard Tableaux without our optimisation and caching strategies?
- Do optimisation and caching strategies result in significant overhead in terms of processing time?
- Do different strategies work better for different case studies / inference tasks? Which mTableaux strategies or combination of strategies work best?

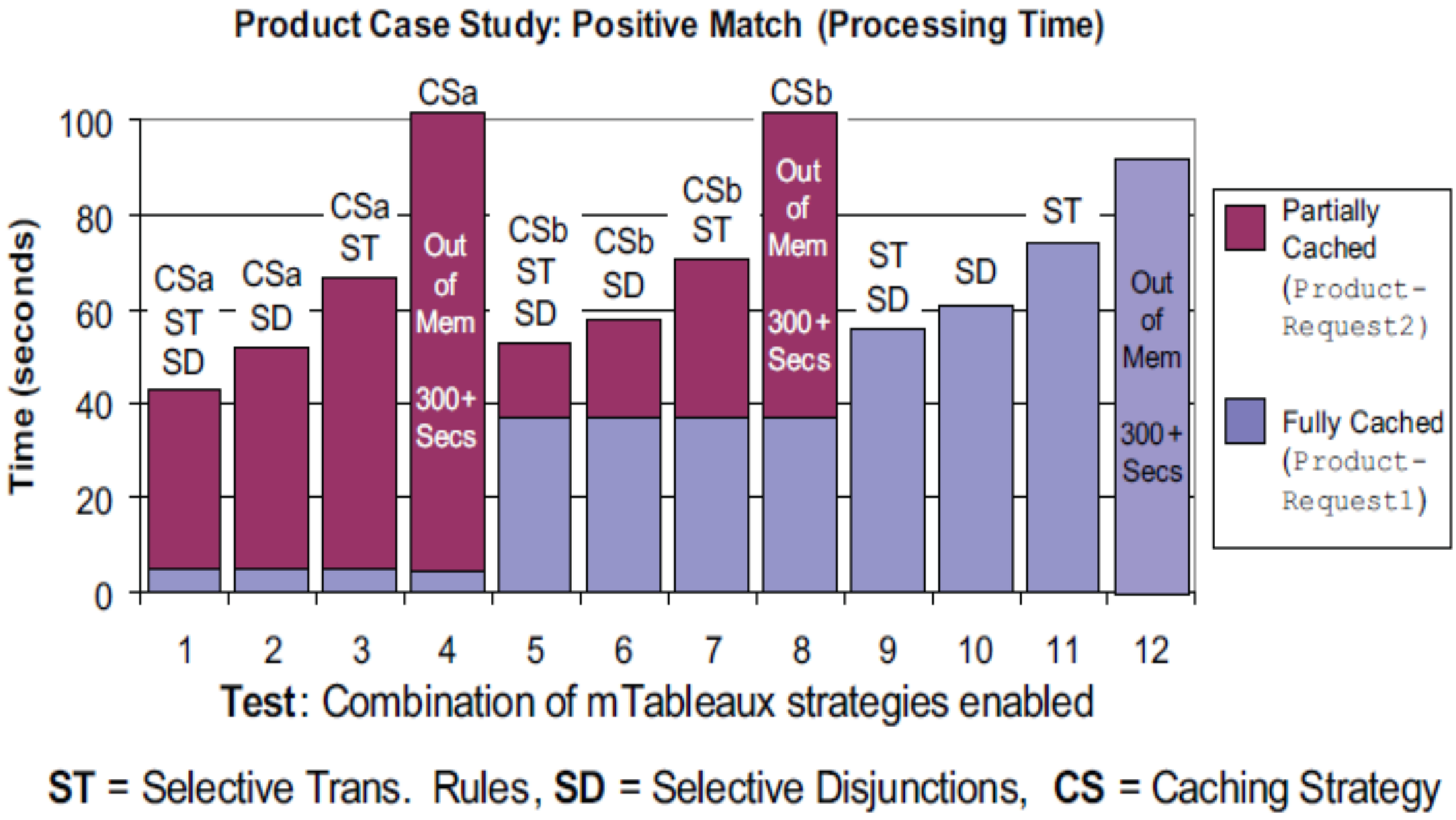
# Evaluation

| Test Number | | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Selective Transformations | | (ST) | ✓ | | ✓ | | ✓ | | ✓ | | ✓ | | ✓ | |
| Selective Disjunctions | | (SD) | ✓ | ✓ | | | ✓ | ✓ | | | ✓ | ✓ | | |
| Cache | Immediate Match | (CSa) | ✓ | ✓ | ✓ | ✓ | | | | | | | | |
| | Re-Order Evaluations | (CSb) | | | | | ✓ | ✓ | ✓ | ✓ | | | | |

# Evaluation — Mobile Devices

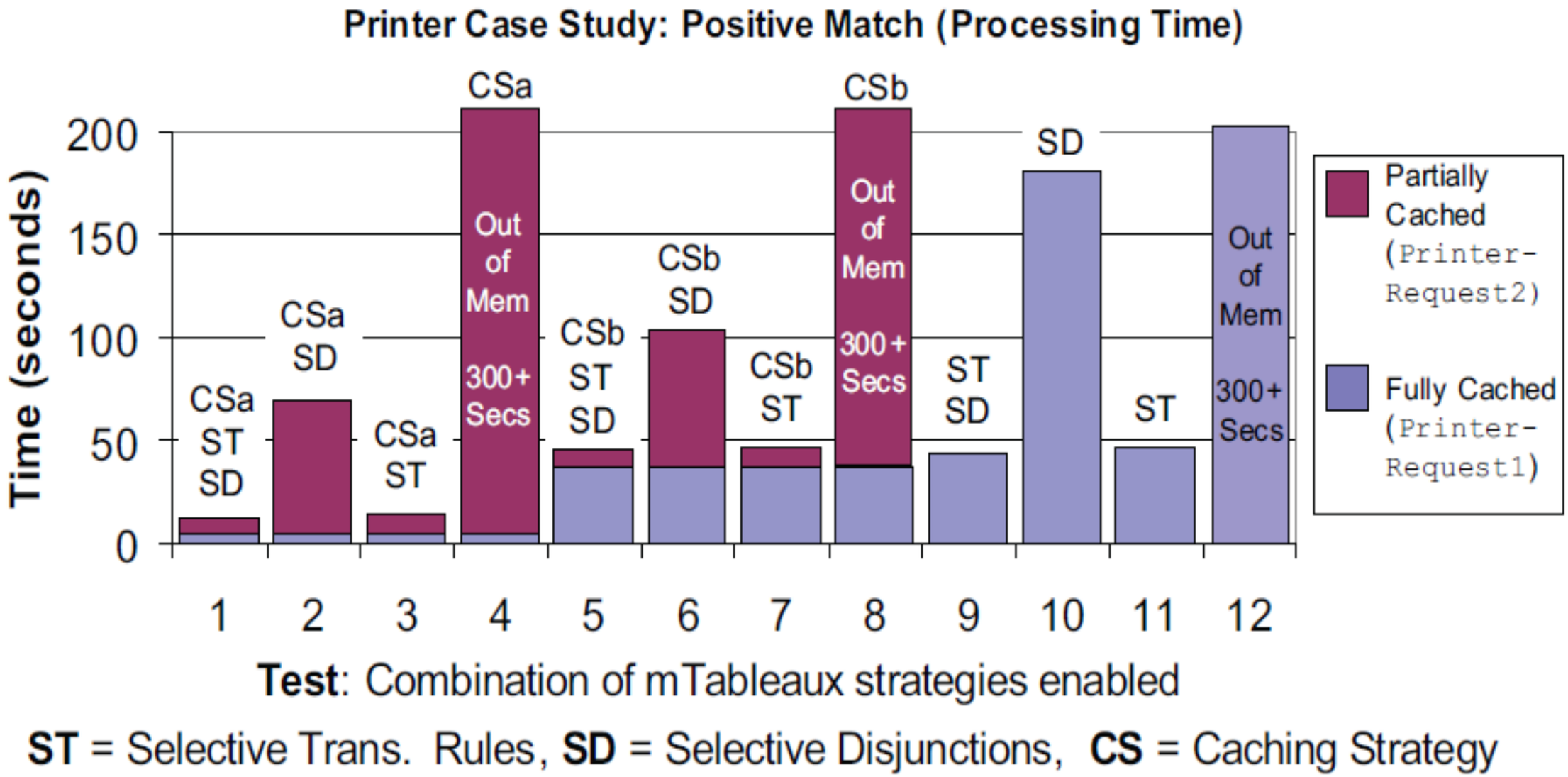
- The  evaluations were performed on a HP iPAQ hx2700 PDA, with Marvell PXA270 624MHz processor, 256MB memory (64MB main memory, 192MB flash ROM) with the Windows Mobile 5.0 operating system.
- Intentional choice of low computational device to stress test the reasoner.
- Currently available HP PDAs include the HP iPAQ 11219 and HP iPAQ 21220, which have 624MHz processors and 64MB and 128MB main memory.
- Our evaluations are performed using the Mysaifu21 Java J2SE Virtual Machine (JVM) - 15MB allocated.
- mTableaux is implemented in the Pellet reasoner with *SHOIN* expressiveness.

# Evaluation

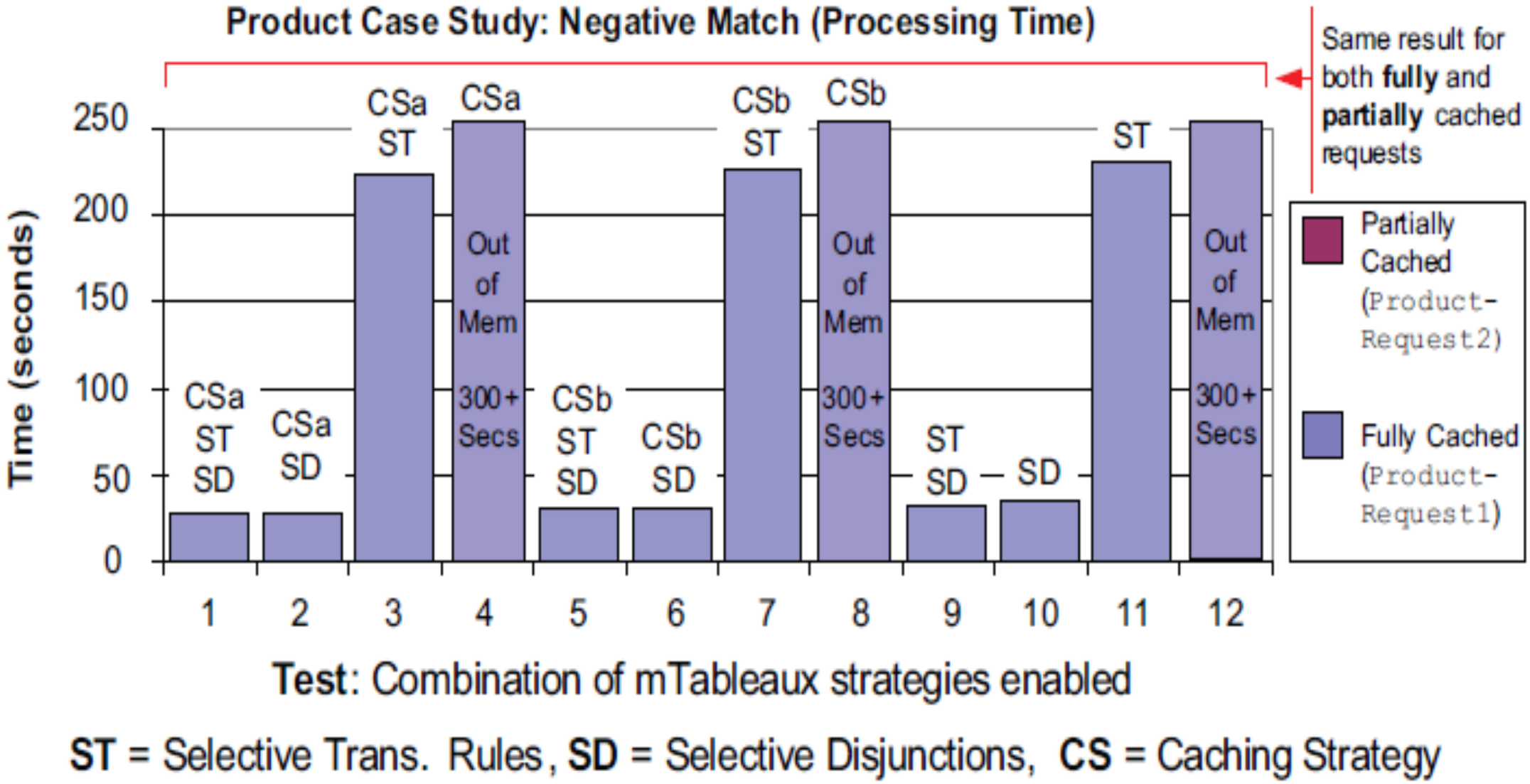



**ST** = Selective Trans. Rules, **SD** = Selective Disjunctions, **CS** = Caching Strategy

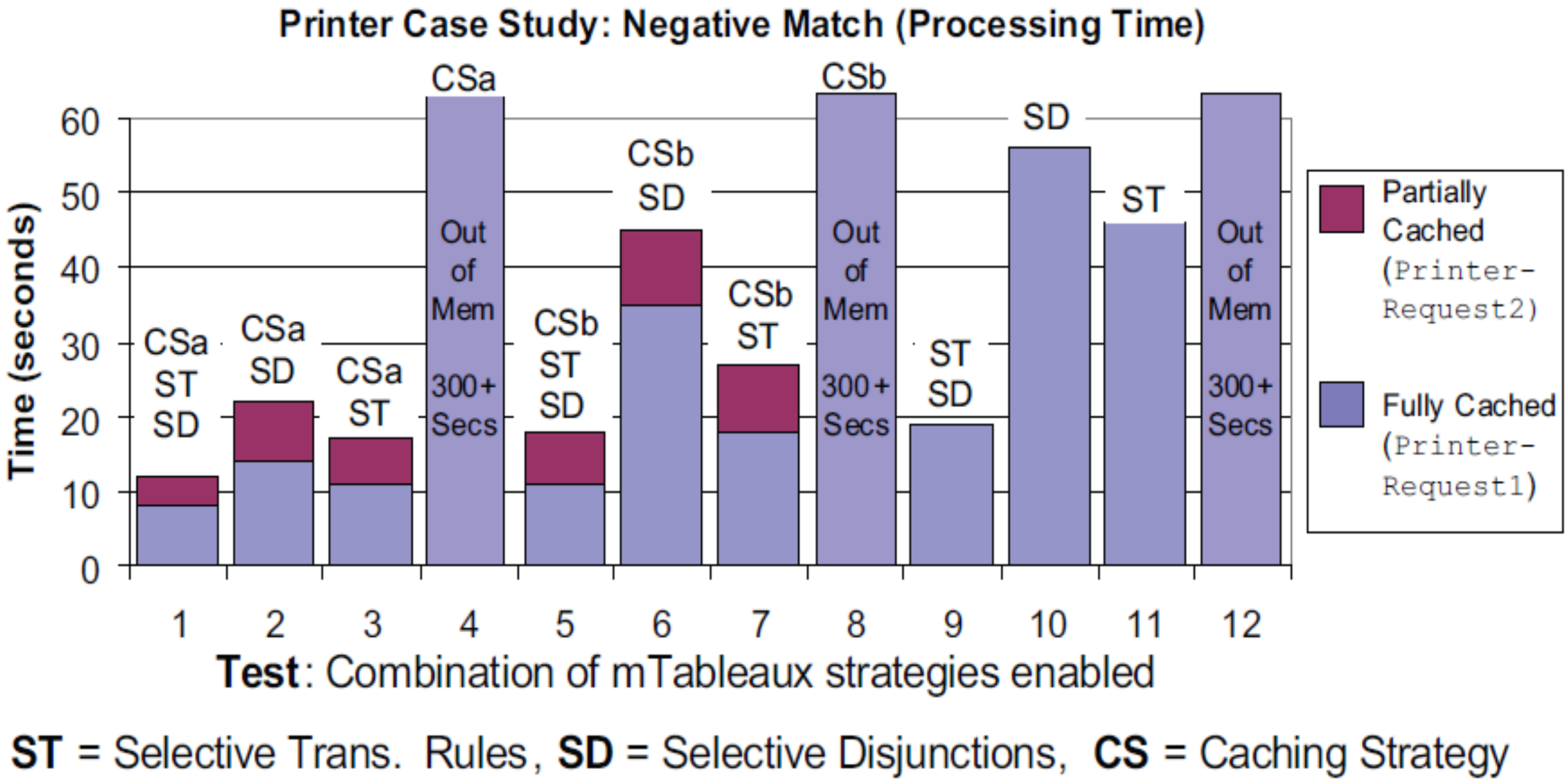
# Evaluation


Printer Case Study: Positive Match (Processing Time)
Time (seconds)
200
150
100
50
0
CSa ST SD
CSa SD
CSa ST
CSa
Out of Mem
300+ Secs
CSb ST SD
CSb SD
CSb
Out of Mem
300+ Secs
CSb ST
ST SD
SD
ST
Out of Mem
300+ Secs
1
2
3
4
5
6
7
8
9
10
11
12
Test: Combination of mTableaux strategies enabled
Partially Cached (Printer-Request2)
Fully Cached (Printer-Request1)
ST = Selective Trans. Rules, SD = Selective Disjunctions, CS = Caching Strategy

# Evaluation

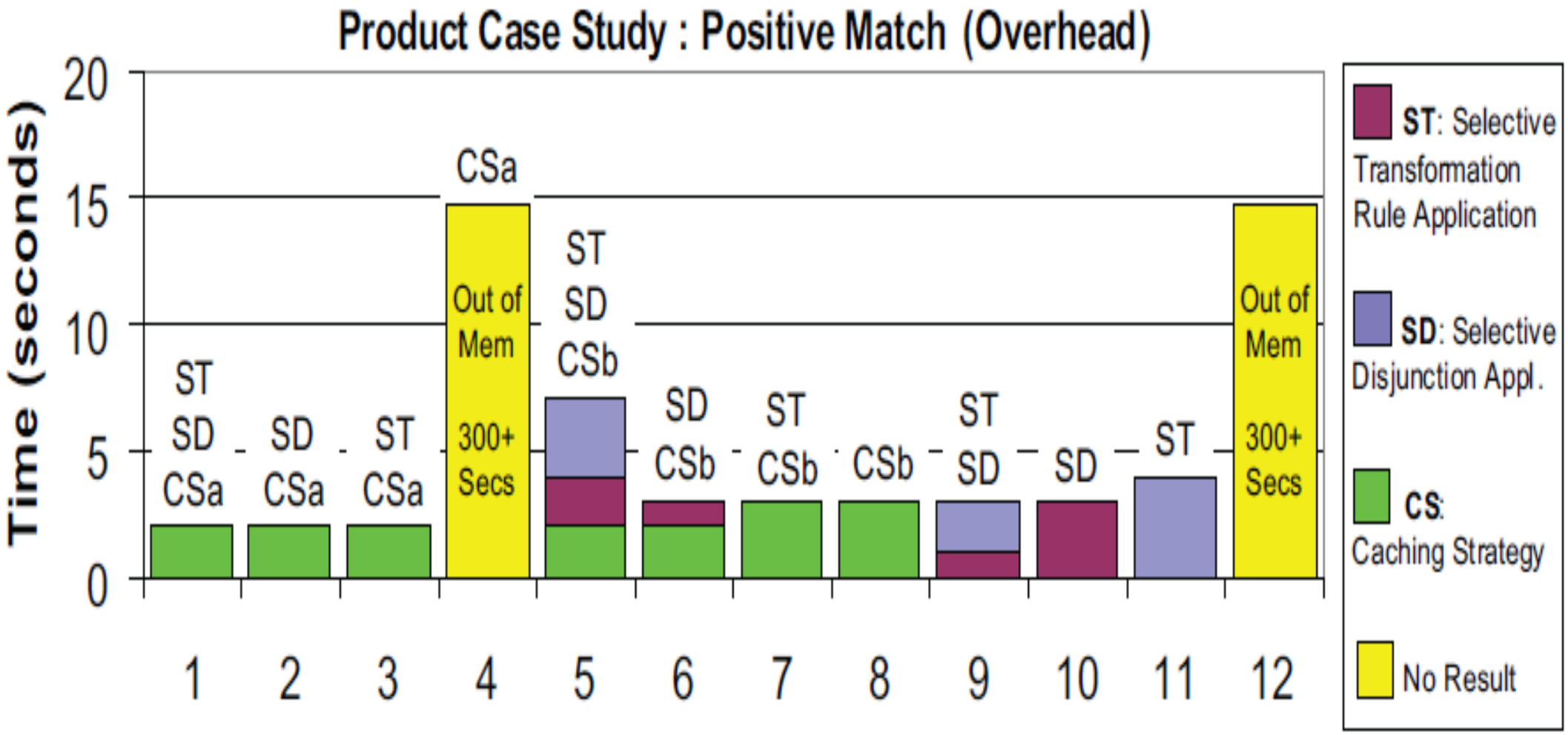

Product Case Study: Negative Match (Processing Time)
Same result for both fully and partially cached requests
Time (seconds)
250
200
150
100
50
0
CSa ST
CSa
CSb ST
CSb
ST
Out of Mem
300+ Secs
CSa ST SD
CSa SD
CSb ST SD
CSb SD
ST SD
SD
Partially Cached (Product-Request2)
Fully Cached (Product-Request1)
1
2
3
4
5
6
7
8
9
10
11
12
Test: Combination of mTableaux strategies enabled
ST = Selective Trans. Rules, SD = Selective Disjunctions, CS = Caching Strategy

# Evaluation

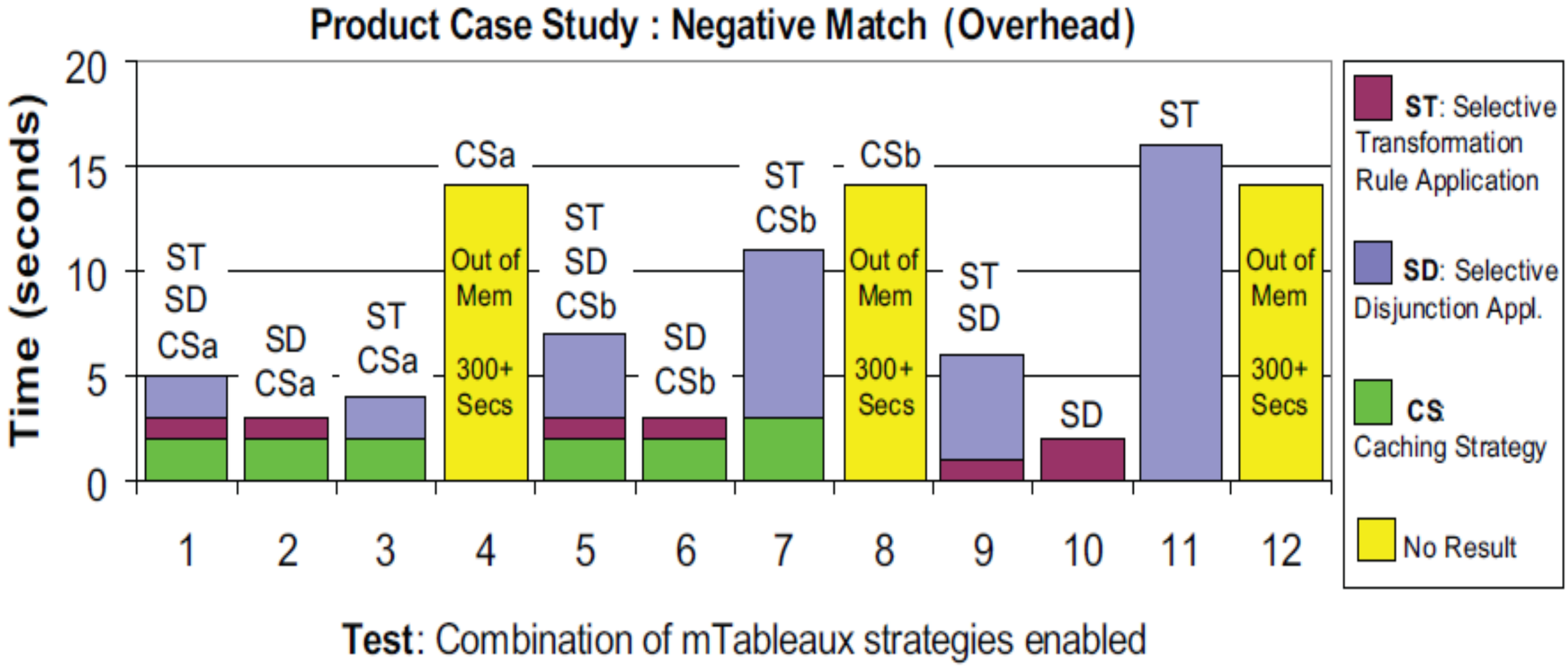



**ST** = Selective Trans. Rules, **SD** = Selective Disjunctions, **CS** = Caching Strategy

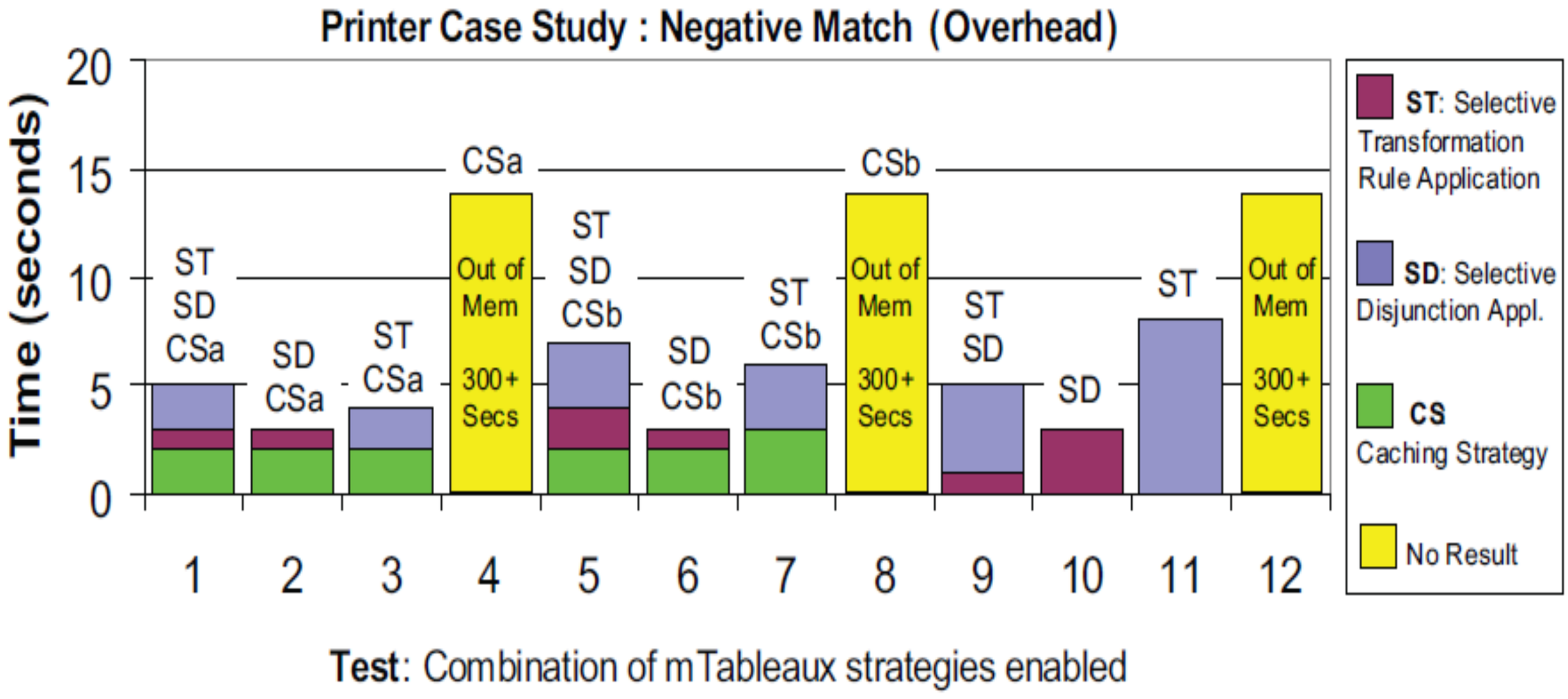
# Evaluation — Overhead of Strategies



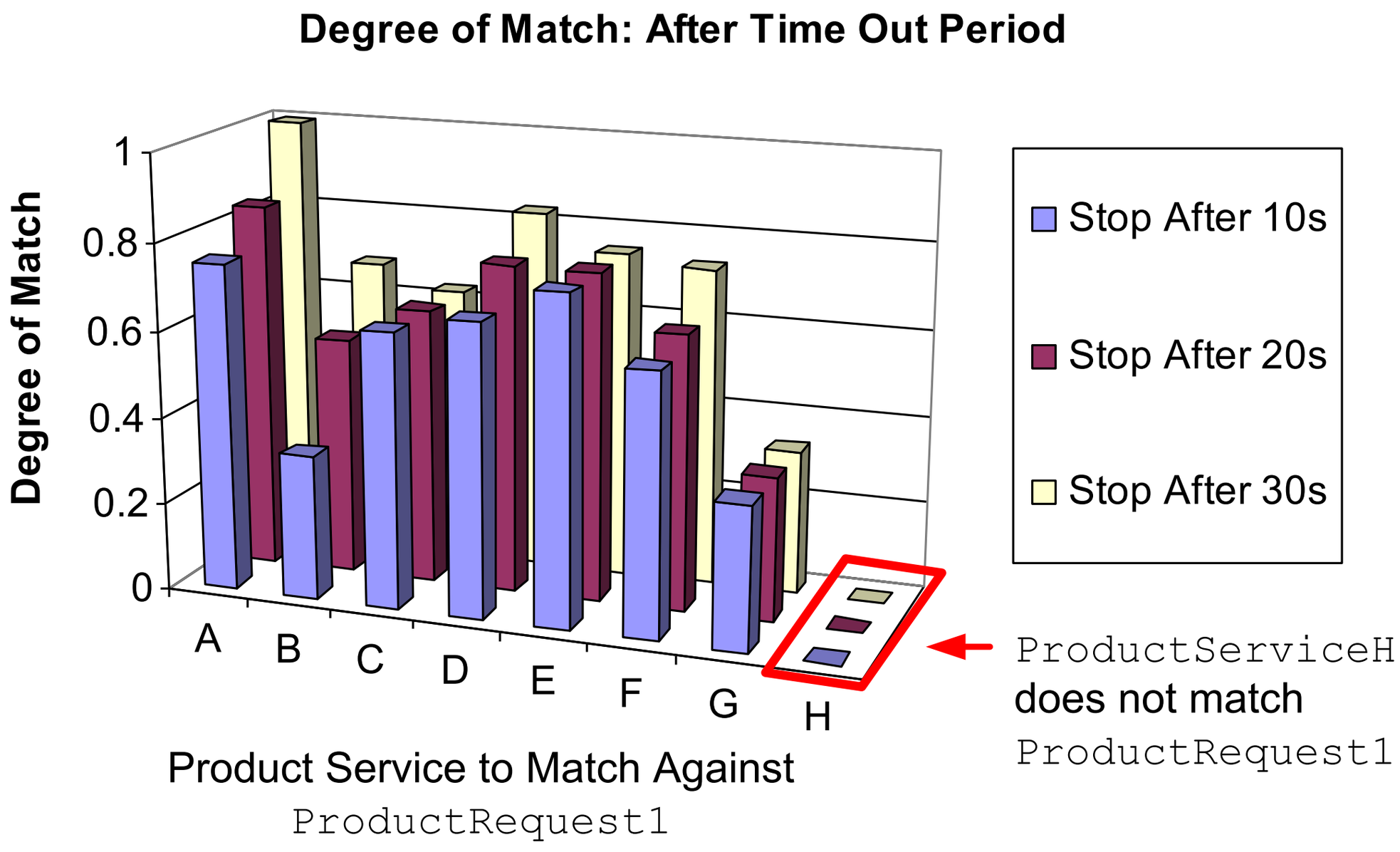
# Evaluation


Product Case Study : Negative Match (Overhead)
Time (seconds)
20
15
10
5
0
ST SD CSa
SD CSa
ST CSa
CSa
Out of Mem 300+ Secs
ST SD CSb
SD CSb
ST CSb
CSb
Out of Mem 300+ Secs
ST SD
SD
ST
Out of Mem 300+ Secs
1
2
3
4
5
6
7
8
9
10
11
12
Test: Combination of mTableaux strategies enabled
ST: Selective Transformation Rule Application
SD: Selective Disjunction Appl.
CS: Caching Strategy
No Result

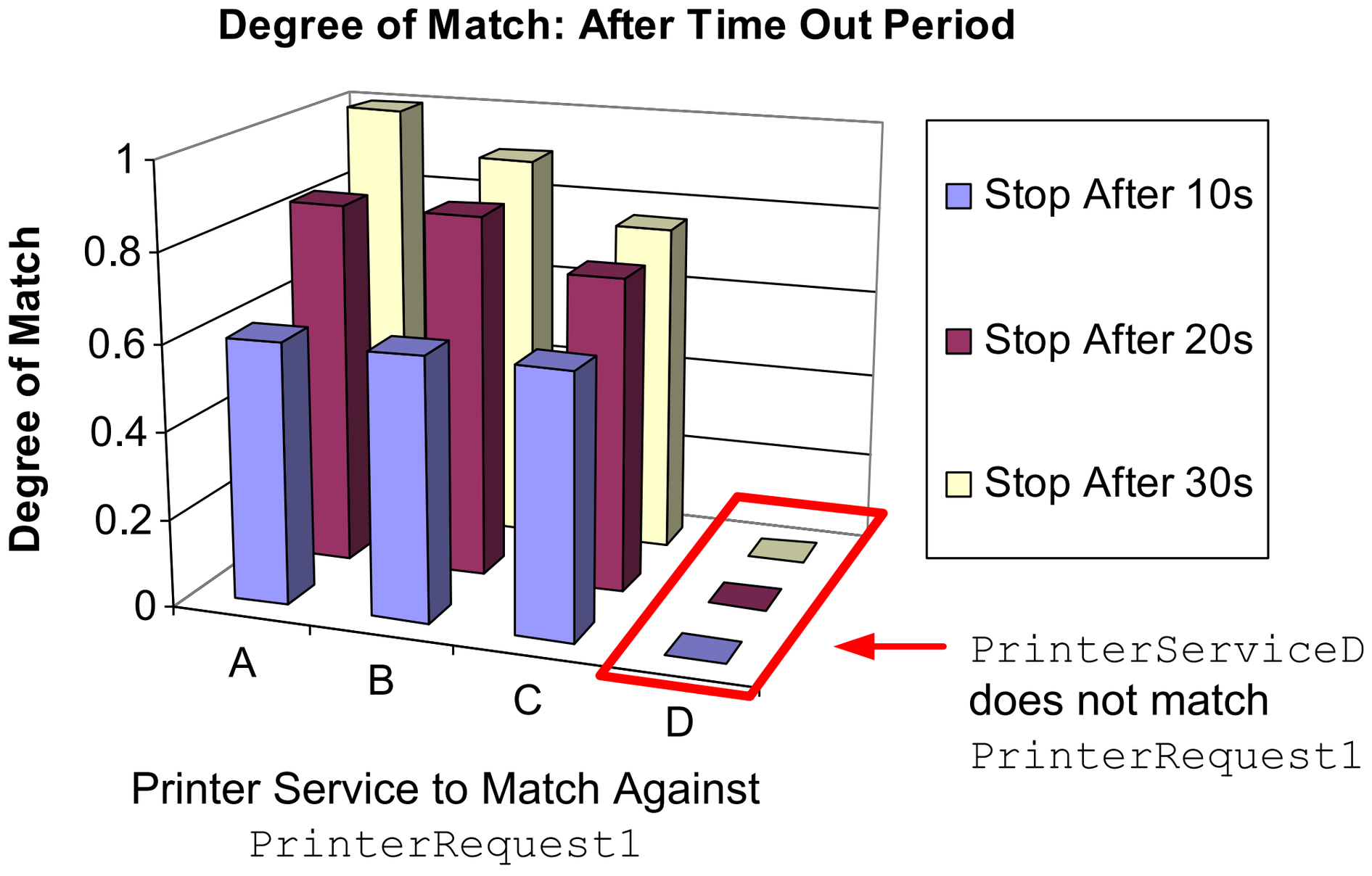
# Evaluation


Printer Case Study : Negative Match (Overhead)
Time (seconds)
20
15
10
5
0
ST SD CSa
SD CSa
ST - CSa
Out of Mem 300+ Secs
ST SD CSb
SD CSb
ST CSb
Out of Mem 300+ Secs
ST SD
SD
ST
Out of Mem 300+ Secs
CSa
CSb
1 2 3 4 5 6 7 8 9 10 11 12
Test: Combination of mTableaux strategies enabled
ST: Selective Transformation Rule Application
SD: Selective Disjunction Appl.
CS Caching Strategy
No Result

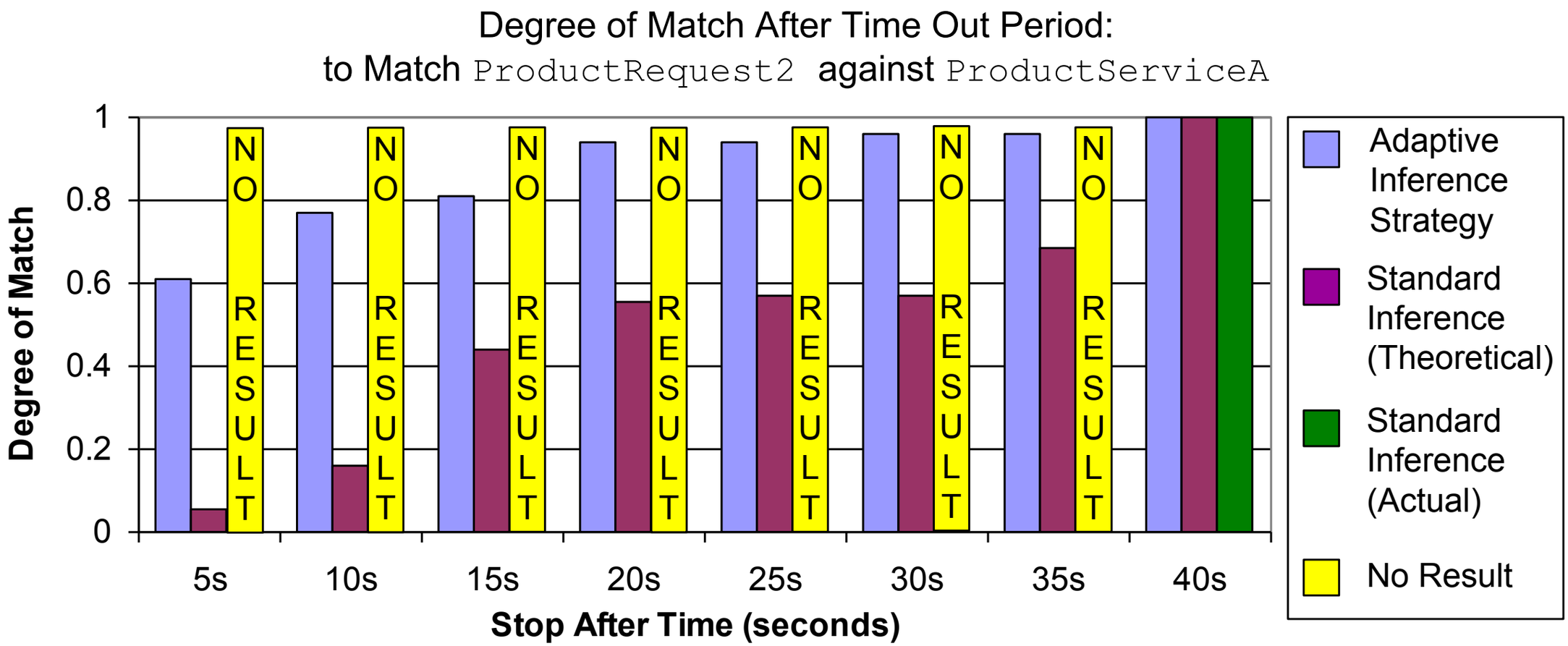
# Adaptive Inference Strategy
# Degree of Match – Product Scenario (after timeout)



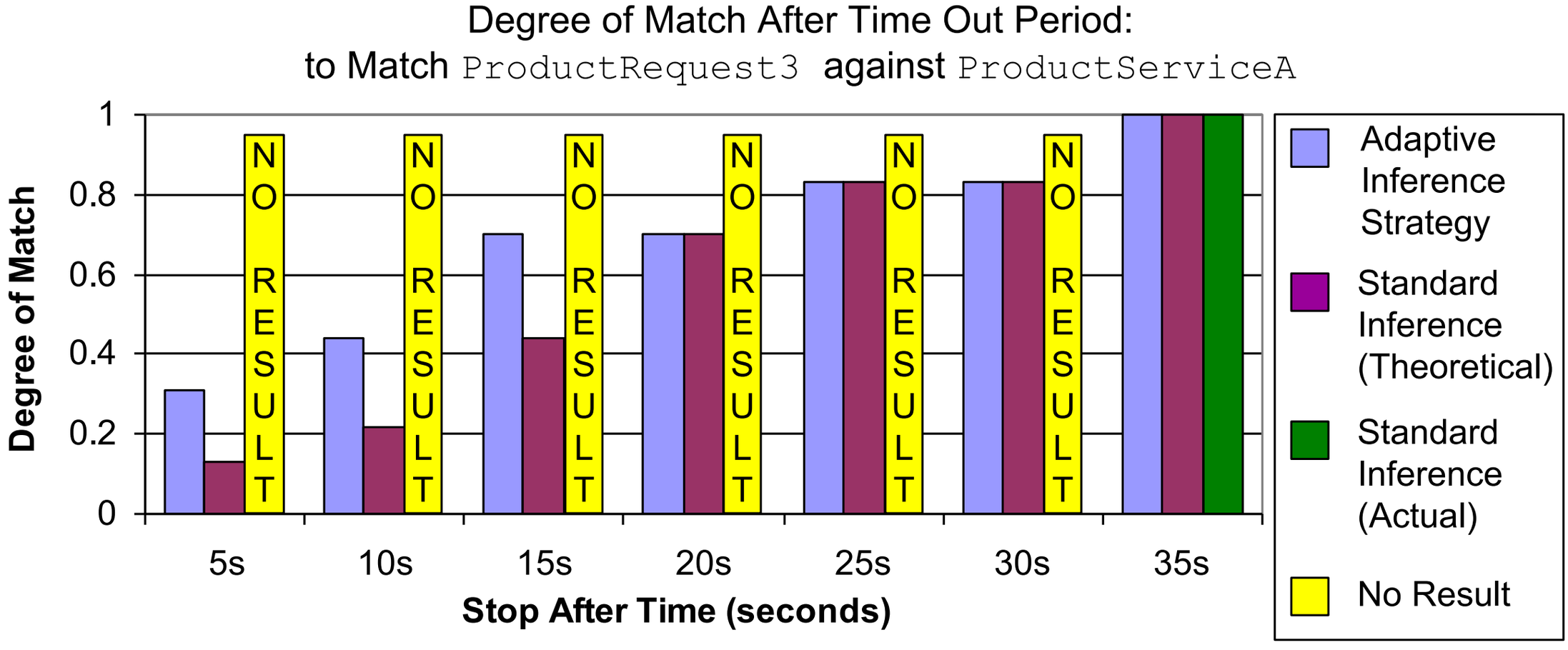
# Adaptive Inference Strategy
# Degree of Match – Printer Scenario (after timeout)



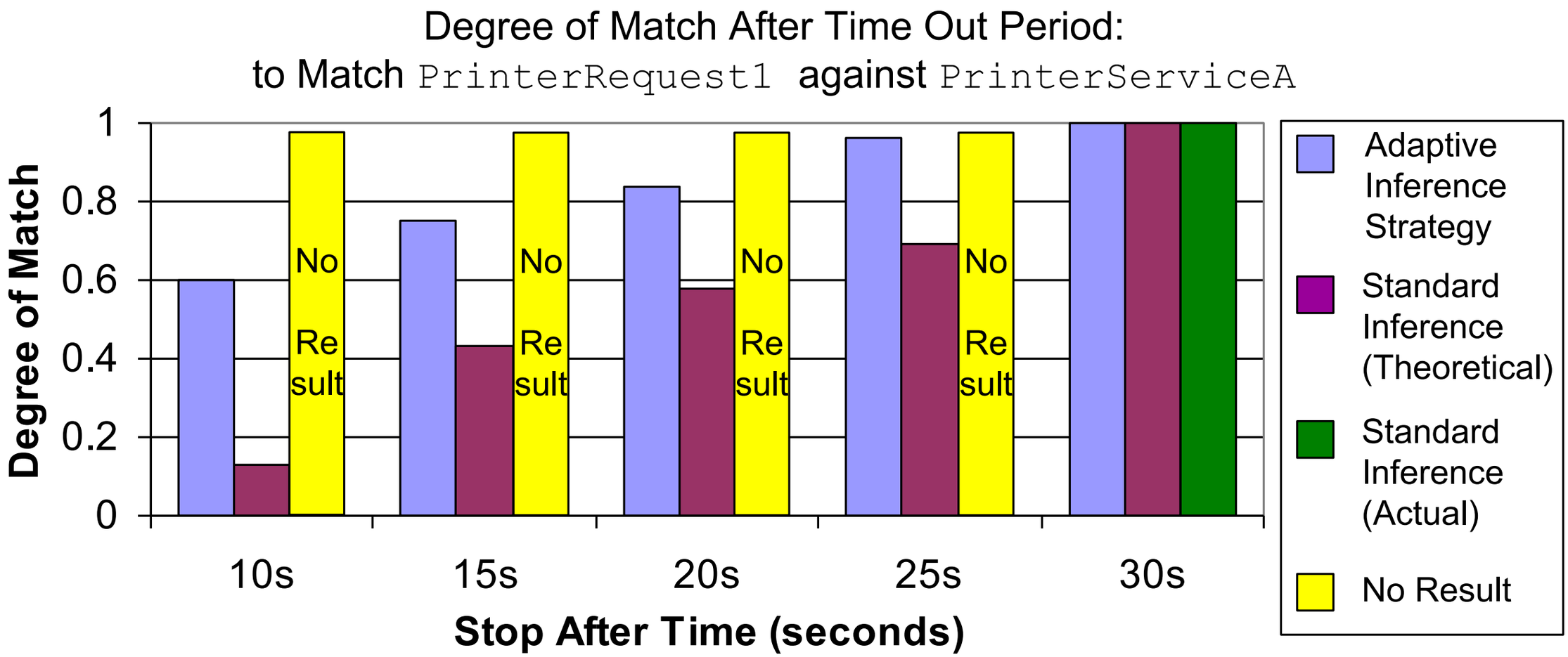
# Adaptive Inference Strategy
# Degree of Match (after timeout) - Product Scenario



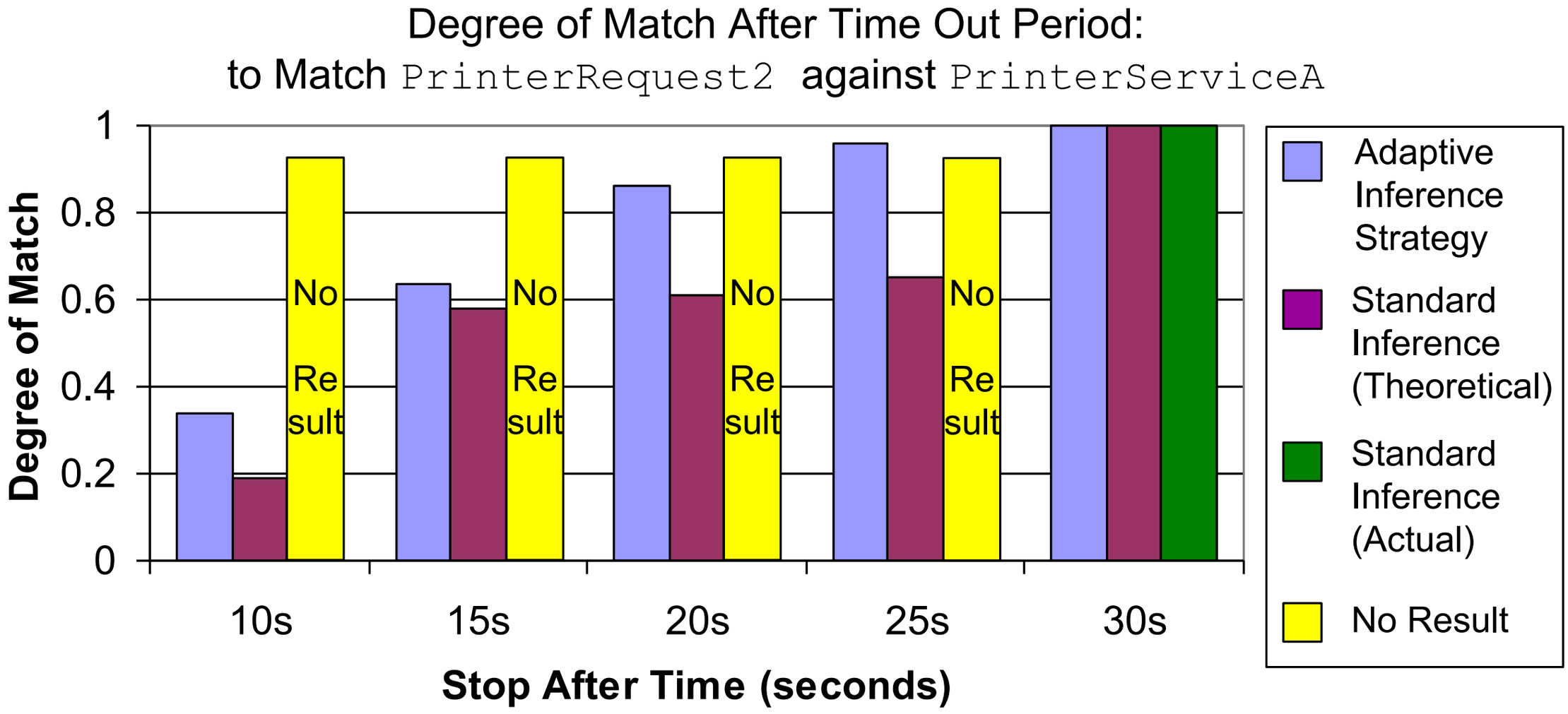
# Adaptive Inference Strategy
## Degree of Match (after timeout) - Product Scenario



# Adaptive Inference Strategy
# Degree of Match (after timeout) - Printer Scenario



# Adaptive Inference Strategy
# Degree of Match (after timeout) - Printer Scenario



# Summary of Evaluation

- mTableaux reduces the size of the inference problem by reducing the number of expansions performed by the reasoner.
- mTableaux successfully enables the completion of a matching task on a small, resource constrained device without exceeding available memory.
- Conversely, without the strategies enabled the task could not be completed.
- mTableaux significantly improves the response time to perform matching when the optimisation and caching strategies were enabled compared to when these were not enabled. In fact without the strategies enabled the task could not be completed.

# Summary of Evaluation

- mTableaux optimisation and caching strategies used together has minimal extra processing overhead.
- mTableaux strategies improved performance for inference checks which both provided a positive and a negative match result.
- The effectiveness of each optimisation strategy depends on the ontology which matching is being performed on.
- The ST optimisation strategy was more effective in improving efficiency in the first case study while SD was more effective in the second case study.
- Utilising all strategies together provided the fastest / most efficient response-time.

# Summary of Evaluation

- mTableaux reduces the size of the inference problem by reducing the number of expansions performed by the reasoner.
- mTableaux successfully enables the completion of a matching task on a small, resource constrained device without exceeding available memory.
- Conversely, without the strategies enabled the task could not be completed.
- mTableaux significantly improves the response time to perform matching when the optimisation and caching strategies were enabled compared to when these were not enabled. In fact without the strategies enabled the task could not be completed.

# Summary of Evaluation

- mTableaux optimisation and caching strategies used together has minimal extra processing overhead.
- mTableaux strategies improved performance for inference checks which both provided a positive and a negative match result.
- The effectiveness of each optimisation strategy depends on the ontology which matching is being performed on.
- The ST optimisation strategy was more effective in improving efficiency in the first case study while SD was more effective in the second case study.
- Utilising all strategies together provided the fastest / most efficient response-time.

# Related Work & Recent Development

# Semantic Reasoners for Mobile Devices

- KRHyper: Kleemann and Sinner (2006)
  - Novel Tableaux reasoner for First Order Logic (FOL) for deployment on resource constrained devices
  - In order to use DL with KRHyper it must be transformed into a set of disjunctive first order logic clauses
  - It implements the standard Tableaux optimisation strategies of backjumping, semantic branching, Boolean constraint propagation, lazy unfolding and absorption used by today's commercial and open source reasoners.

# Semantic Reasoners for Mobile Devices

- KRHyper: Kleemann and Sinner (2006)
  - Performance comparisons show that KRHyper provides performance which is compariable to RacerPro but tends to provide the same or better response time for test cases which contained 10 or fewer subsumption checks, and did not perform as well for larger tests.
  - All tests had short branch sizes with less than 25 Tableaux proof steps, because the test ontology was not deeply nested.
  - KRHyper still exhausts all memory when the reasoning task becomes too large for a small device to handle and fails to provide any result.

# Semantic Reasoners for Mobile Devices

- Gu et al. (2007)
  - Framework which provides an RDF/OWL parser, reasoner and sRDQL query engine for information matching
  - Example: a shopping assistance application which uses a user's context to provide suggestions to the user about products which may suit their needs based on previous usage patterns.
  - Runs on the user's mobile device using J2ME
- This framework provides acceptable performance by supporting OWL Lite
- The reasoner in the framework uses a forward chaining approach which supports rule based reasoning

# Semantic Reasoners for Mobile Devices

- Ruta et al. (2008a,b)
  - supports distance based, ranked, matching of requests to services using a DL mobile reasoner implemented in Java 2 Micro Edition (J2ME)
  - supports short range (bluetooth) ad-hoc networks
  - Achieves acceptable performance by restricting the OWL-DL language to a subset
  - Ontology structure is constrained so that it can be reduced to set comparisons for reducing computational overhead
- Ruta et al. (2012)
  - Mini-ME: an $\mathcal{ALCN}$ reasoner Android
  - Improved memory footprint compared to previous version

# Semantic Reasoners for Embedded Devices

- An ontology reasoner on Programmable Logic Controllers: Grimm (2012)
  - For $\mathcal{EL}^+$: without concept disjointness & nominals (individuals)
  - Used for industrial diagnostics
- Programmable logic controllers:
  - Widely used in industrial automation
  - Are resource-constrained:
    - Fixed-length execution cycles
    - 4MB memory (S7-300, S7-400)
- Reasoning strategies
  - A compact representation for axioms
  - An interruption-safe implementation of completion rules

# Delta-Reasoner

- Delta-Reasoner: Motik, Horrocks & Kim (2012)
  - A reasoner to support context-awareness
  - Incremental OWL 2 RL reasoning
  - Continuous conjunctive query answering
- Performance evaluated on a laptop
  - 3 ontologies
  - Axioms added & removed

| | TBox axioms | Class assertions | Property assertions | DL expressivity |
|---|---|---|---|---|
| VICODI | 223 | 33,238 | 82,943 | $\mathcal{ALHI}(\mathbf{D})$ |
| SEMINTEC | 219 | 17,941 | 47,299 | $\mathcal{ALHIF}$ |
| LUBM | 93 | 18,128 | 82,415 | $\mathcal{ALEHI}^{+}(\mathbf{D})$ |

| | Loading | Initial reasoning | Incremental reasoning |
|---|---|---|---|
| VICODI | 2127 | 2820 | 156 |
| SEMINTEC | 1048 | 1123 | 157 |
| LUBM | 2597 | 818 | 135 |

# ELK on Android

- Porting ELK on Android 4.2: Kazakov & Klinov (2013)
  - Supports OWL 2 EL reasoning
- Evaluation
  - 5 OWL 2 EL ontologies: ChEBI, EMAP, Fly Anatomy, Gene Ontology & EL-GALEN
  - *Acceptable* performance
  - Still magnitudes slower than on desktop computers

| | | Google Nexus 4 | | | | PC | | |
|---|---|---|---|---|---|---|---|---|
| Ontology | Workers | Load./Index. | Classif. | LI Ratio | Memory | Load./Index. | Classif. | LI Ratio |
| ChEBI | 1 | 31,370 | 207,020 | 13 | 67 | 351 | 1,055 | 25 |
| ChEBI | 2 | 29,423 | 160,334 | 16 | 72 | 323 | 715 | 31 |
| ChEBI | 3 | 32,213 | 148,369 | 18 | 72 | 337 | 611 | 36 |
| ChEBI | 4 | 32,443 | 147,868 | 18 | 68 | 324 | 646 | 33 |
| ChEBI | 5 | 32,900 | 114,054 | 22 | 65 | 362 | 570 | 39 |
| ChEBI | 6 | 29,997 | 107,033 | 22 | 72 | 341 | 597 | 36 |

# What about Other Reasoners?

- Comparison analysis of modern reasoners on Android: Yus et al. (2013)
  - JFaCT, CB, HermiT, Pellet

| | | JFact | CB | HermiT | Pellet |
|---|---|---|---|---|---|
| Pizza | PC | 0.37 | $0^{\Diamond}$ | 0.57 | 0.97 |
| | Android1 | 4.90 | $0^{\Diamond}$ | 14.88 | 33.22 |
| | Android2 | 3.42 | $0^{\Diamond}$ | 10.43 | 20.77 |
| Wine | PC | 10.39 | $0^{\Diamond}$ | 6.54 | 2.22 |
| | Android1 | 2196.05 | $0^{\Diamond}$ | 511.97 | 194.12 |
| | Android2 | 1609.32 | $0^{\Diamond}$ | 361.38 | 131.80 |
| DBpedia | PC | UDT! | 0 | 0.10 | 1.39 |
| | Android1 | UDT! | 0 | 8.87 | 115.30 |
| | Android2 | UDT! | 0 | 5.13 | 63.15 |
| GO | PC | 7.77 | 0.11 | 1.56 | 1.96 |
| | Android1 | OOM! | 1.95 | OOM! | OOM! |
| | Android2 | 435.60 | 1.47 | 487.98 | 83.97 |
| NCI | PC | 2.61 | 0.24 | 2.23 | 4.24 |
| | Android1 | OOM! | 3.31 | OOM! | OOM! |
| | Android2 | OOM! | 2.69 | 2020.48 | OOM! |

# Semantic Reasoners for Mobile Devices

- Typical Approaches — restricting the OWL language to a subset to reduce the resource demands on the device
- Works to a degree
- Still Magnitudes slower than on a desktop computer!
- More research needed!

# Future Directions

- Implementation as a generic plug-in for reasoning on Android platform
- Adaptation to other constraints – resources, confidence
- Reasoning time complexity is important…
  - Predictive metrics
- Distributed reasoning – mobile and cloud

# Thank you!

# Questions?

# References


- Bener, A. B., Ozadali, V. and Ilhan, E. S. (2009). Semantic matchmaker with precondition and effect matching using SWRL, *Expert Systems with Applications, Pergamon Press* **36**(5): 9371 – 9377.
- Brambilla, M., Celino, I., Ceri, S., Cerizza, D., della Valle, E., Facca, F. and Tziviskou, C. (2006). Improvements and future perspectives on web engineering methods for automating web services mediation, choreography and discovery: SWS-challenge phase III, *3rd Workshop of the Semantic Web Service Challenge*, Athens, GA, USA.
- Chen, H., Finin, T. and Joshi, A. (2004). Semantic web in the context bro- ker architecture, *2nd International Conference on Pervasive Computer and Communications (PerCom 04)*, IEEE, Orlando, Florida, pp. 277 – 286.
- Chakraborty, D., Perich, F., Avancha, S. and Joshi, A. (2001). Dreggie: Semantic service discovery for m-commerce applications, *Workshop on Reliable and Secure Applications in Mobile Environment*, New Orleans, Louisiana, USA.
- Kuster, U. and König-Ries, B. (2008). Semantic service discovery with DIANE service descriptions, *Semantic Web Services Challenge*, Vol. 8, Springer- Verlag, pp. 199 – 216.

# References


- Masuoka, R., Parsia, B. and Labrou, Y. (2003). Task computing - the semantic web meets pervasive computing, *International Semantic Web Conference (ISWC '03)*, Vol. 2870, Springer-Verlag, pp. 866 – 881.
- Luo, J., Montrose, B. and Kang, M. (2005). An approach for semantic query processing with UDDI, *On the Move to Meaningful Internet Systems*, Vol. 3762, Springer-Verlag, pp. 89 – 98.
- Srinivasan, N., Paolucci, M. and Sycara, K. (2005). Semantic web service discovery in the OWL-S IDE, *39th International Conference on System Sci- ences*, Vol. 6, IEEE, Hawaii, USA, pp. 109 – 119.
- Stuckenschmidt, H. and Kolb, M. (2008). Partial matchmaking for complex product and service descriptions, *Multikonferenz Wirtschaftsinformatik (MKWI '08)*, GITO-Verlag, Munich, Germany.
- Sycara, K., Widoff, S., Klusch, M. and Lu, J. (2002). LARKS: Dynamic match-making among heterogeneous software agents in cyberspace, *Autonomous Agents and Multi-Agent Systems, Kluwer Academic* **5**(2): 173 – 203.

# References


- Baader, F., Brandt, S., & Lutz, C. (2005, July). Pushing the EL envelope. In *IJCAI* (Vol. 5, pp. 364-369).
- Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. and Patel-Schneider, P. F. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*, ISBN: 0521781760, Cambridge University Press.
- Baader, F., Lutz, C., & Suntisrivaraporn, B. (2006). CEL—a polynomial-time reasoner for life science ontologies. *Automated Reasoning* (pp. 287-291). Springer Berlin Heidelberg.
- Baader, F. and Sattler, U. (2001). An overview of tableau algorithms for description logics, *Studia Logica* **69**(1): 5 – 40.
- Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., & Rosati, R. (2005, July). DL-Lite: Tractable description logics for ontologies. *AAAI* (Vol. 5, pp. 602-607).
- Dentler, K., Cornet, R., ten Teije, A., & de Keizer, N. (2011). Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semantic Web*, *2*(2), 71-87.

# References


- Grimm, S., Watzke, M., Hubauer, T., & Cescolini, F. (2012). Embedded EL+ Reasoning on Programmable Logic Controllers. *ISWC 2012* (pp. 66-81). Springer Berlin Heidelberg.
- Grosof, B. N., Horrocks, I., Volz, R., & Decker, S. (2003, May). Description logic programs: Combining logic programs with description logic. *WWW* (pp. 48-57).
- Gu, T., Kwok, Z., Koh, K. K. and Pung, H. K. (2007). A mobile frame- work supporting ontology processing and reasoning, *2nd Workshop on Requirements and Solutions for Pervasive Software Infrastructure (RSPS)*, Innsbruck, Austria.
- Haarslev, V., & Möller, R. (2001). Description of the RACER System and its Applications. *Description Logics*, *49*.
- Horrocks, I., Kutz, O., & Sattler, U. (2006). The Even More Irresistible SROIQ. *KR*, *6*, 57-67.
- Horrocks, I., & Patel-Schneider, P. F. (2003). Reducing OWL entailment to description logic satisfiability. *ISWC 2003* (pp. 17-29). Springer Berlin Heidelberg.

# References


- Kang, Y. B., Li, Y. F., & Krishnaswamy, S. (2012). Predicting reasoning performance using ontology metrics. *ISWC 2012* (pp. 198-214). Springer Berlin Heidelberg.
- Kazakov, Y. (2009, July). Consequence-Driven Reasoning for Horn SHIQ Ontologies. *IJCAI* (Vol. 9, pp. 2040-2045).
- Kazakov, Y., & Klinov, P. Experimenting with ELK Reasoner on Android. In *2nd OWL Reasoner Evaluation Workshop (ORE 2013)* (p. 68).
- Kazakov, Y., Krötzsch, M., & Simančík, F. (2011). Concurrent Classification of\ mathcal {EL} Ontologies. *ISWC 2011* (pp. 305-320). Springer Berlin Heidelberg.
- Kleemann, T. and Sinner, A. (2006). User profiles and matchmaking on mo- bile phones, *INAP'06*, Springer- Verlag, Fukuoka, Japan, pp. 135 – 147.
- Motik, B., Horrocks, I., & Kim, S. M. (2012, April). Delta-reasoner: a semantic web reasoner for an intelligent mobile platform. WWW 2012, (pp. 63-72). ACM.
- Lawley, M. J., & Bousquet, C. (2010). Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. *6th Australasian Ontology Workshop (IAOA'10),* pp. 45-49.

# References


- Ren, Y., Pan, J. Z., & Zhao, Y. (2010). Soundness Preserving Approximation for TBox Reasoning. *AAAI* (pp. 351-356).
- Ruta, M., Noia, T. D., Sciascio, E. D. and Scioscia, F. (2008b). A semantic-enabled mobile directory service for RFID-based logistics applications, *ICEBE'08*, IEEE, pp. 333 – 340.
- Ruta, M., Scioscia, F., Di Sciascio, E., Gramegna, F., & Loseto, G. Mini-ME: the Mini Matchmaking Engine. *OWL Reasoner Evaluation Workshop (ORE 2012)* (Vol. 858, pp. 52-63).
- Schmidt-Schauß, M., & Smolka, G. (1991). Attributive concept descriptions with complements. *Artificial intelligence*, *48*(1), 1-26.
- Shearer, R., Motik, B., & Horrocks, I. (2008, October). HermiT: A Highly-Efficient OWL Reasoner. In *OWLED* (Vol. 432).
- Simančík, F., Kazakov, Y., & Horrocks, I. (2011, July). Consequence-based reasoning beyond Horn ontologies. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Two* (pp. 1093-1098). AAAI Press.

# References


- Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *JWS*, *5*(2), 51-53.
- Steller, L., & Krishnaswamy, S. (2008). Pervasive Service Discovery: mTableaux Mobile Reasoning. *I-Semantics. Graz, Austria*.
- Steller, L., & Krishnaswamy, S. (2009, March). Efficient mobile reasoning for pervasive discovery. *SAC*, 1247-1251. ACM.
- ter Horst, H. J. (2005). Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *JWS*, *3*(2), 79-115.
- Tsarkov, D., & Horrocks, I. (2006). FaCT++ Description Logic Reasoner: System Description. *Automated reasoning,* 4130, 292–297.
- Yus, R., Bobed, C., Esteban, G., Bobillo, F., & Mena, E. (2013). Android goes semantic: DL reasoners on smartphones. In *2nd OWL Reasoner Evaluation Workshop (ORE 2013)* (p. 46).