



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. Strategies & systems for mobile semantic reasoning
4. Future Directions & Discussions

Mobile Semantic Reasoning:

Why ?

Mobile Semantic Web

What is happening?

- **Proliferation** of Mobile Devices
- **Pervasive** Connectivity
- **Phenomenal** increase in Mobile Content

What is it leading to?

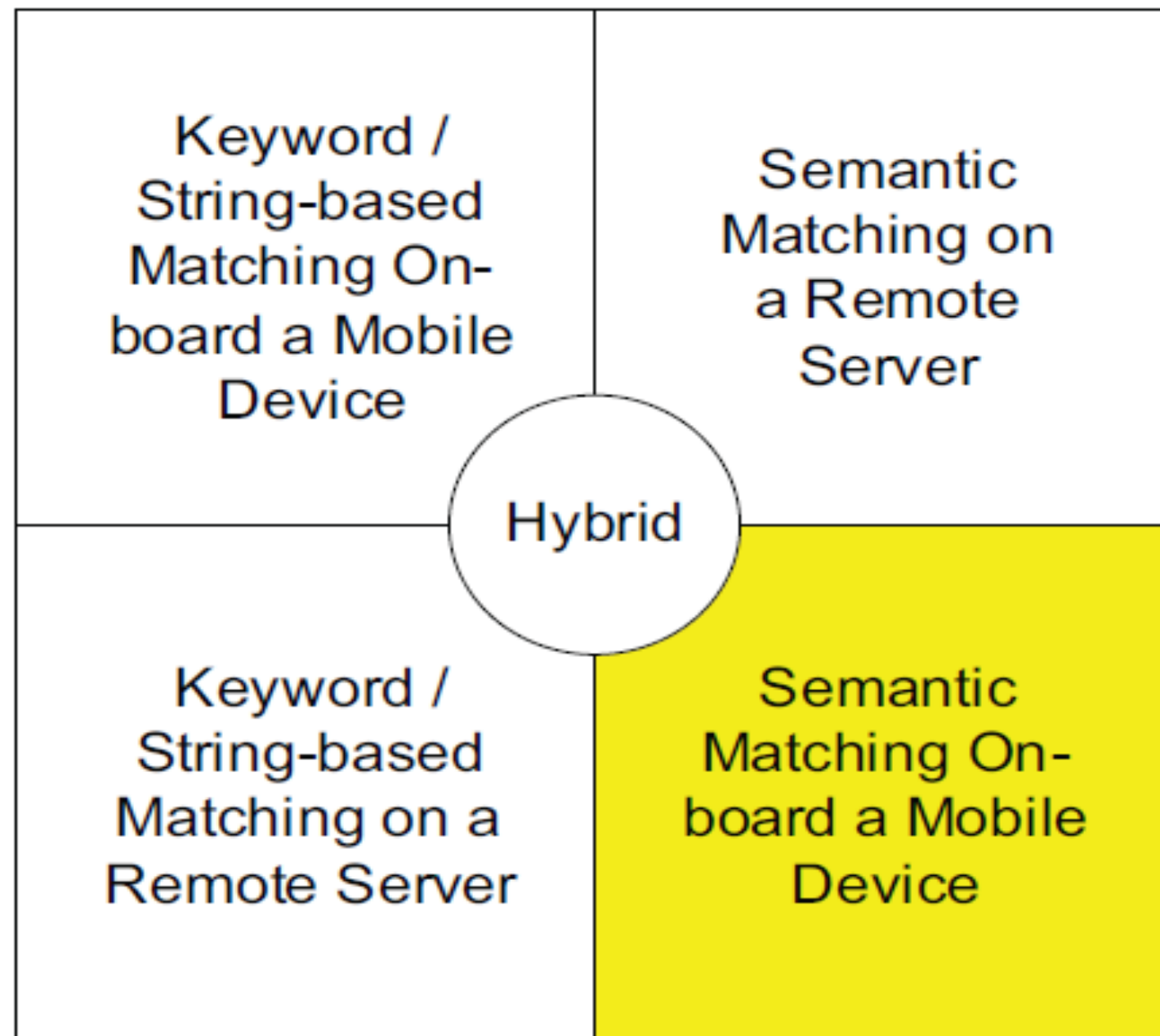
- **Monetisation** of Mobile Data and Content
 - Mobile Crowdsourcing & Crowdsensing
- **Context-Aware** Information & Service Delivery
- **KYC** – Know Your Customer

Why Mobile Semantic Web ?

- Managing Data & Content On-Board Devices
 - Semantic Search on Mobile Devices

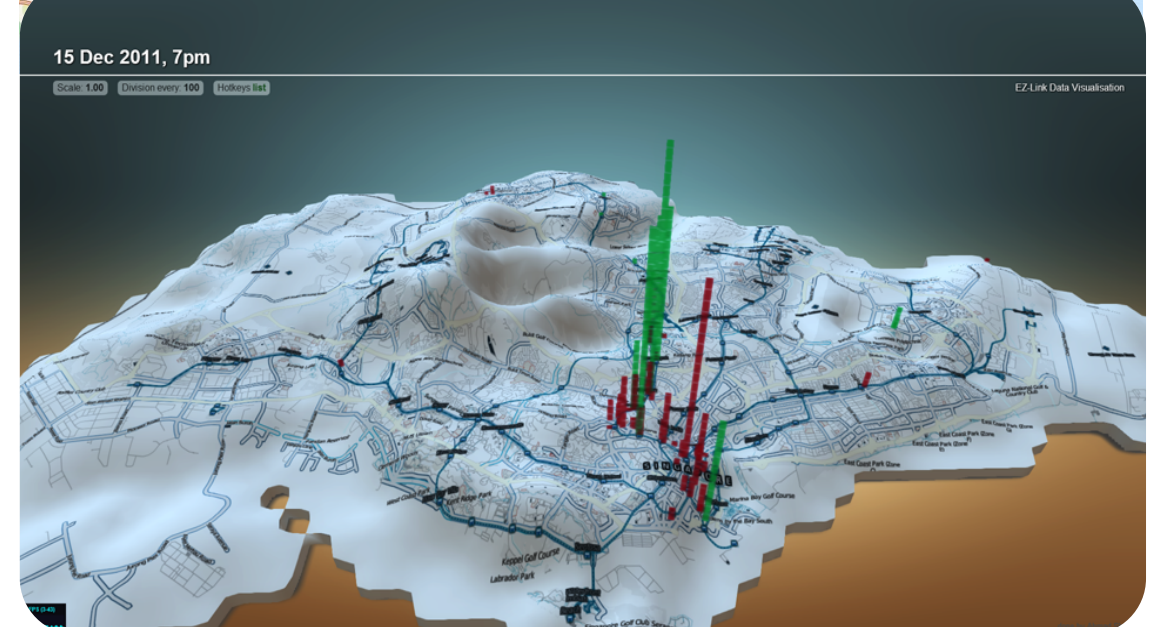
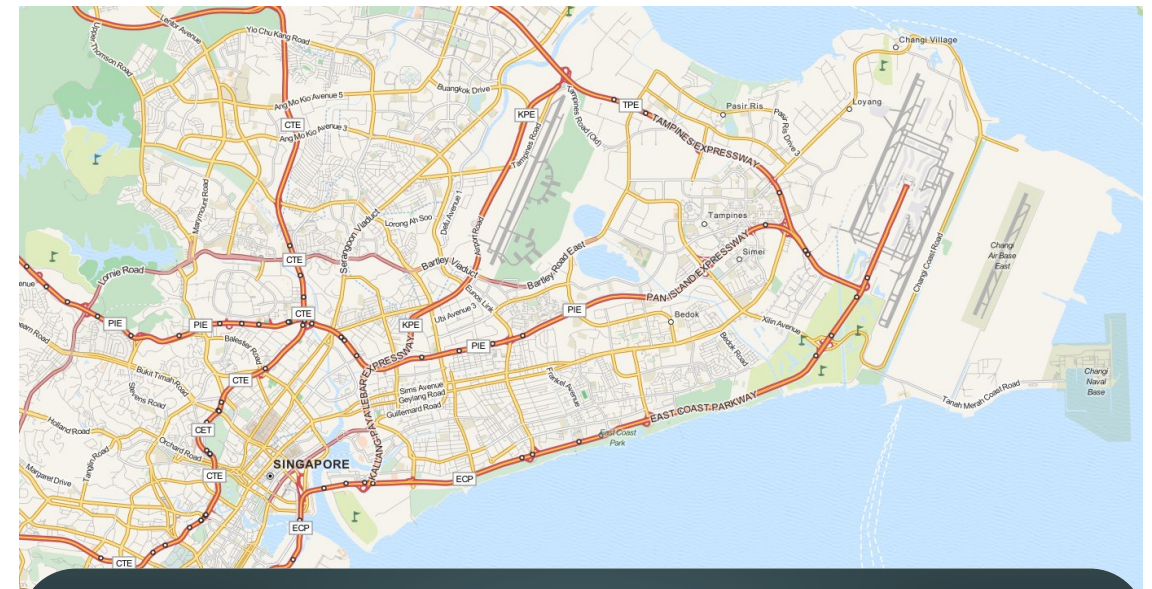
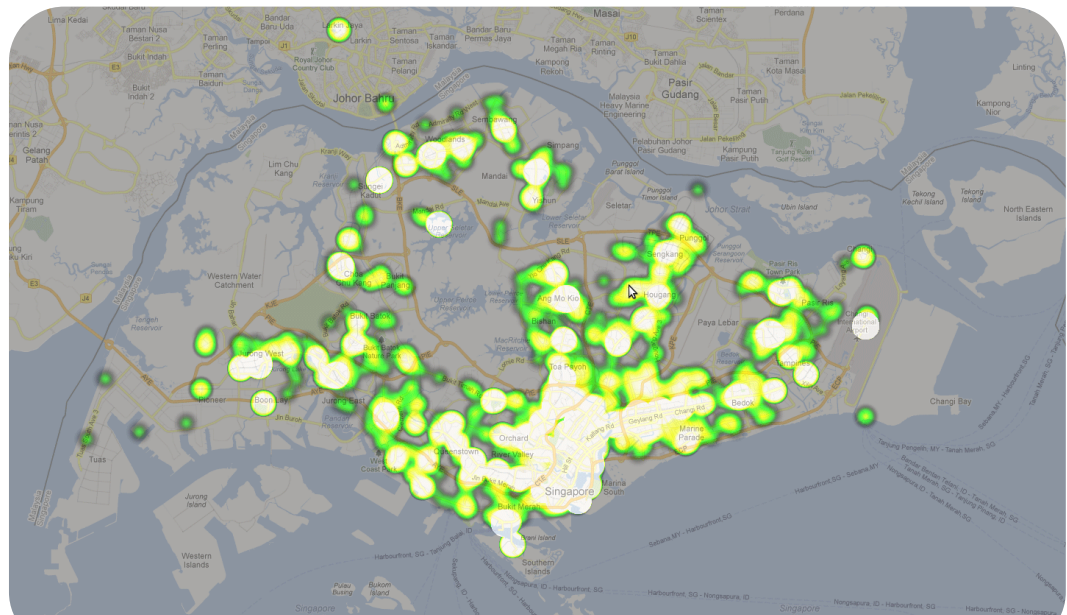


Service Matching in Mobile Environments



Why Mobile Semantic Web ?

- Semantics of Locations & Trajectories
 - Can we infer and manage location semantics on-board devices ?



Why Mobile Semantic Web ?

- The Tell-Tale Phone
 - KYC & Mobile Users

Where is the user now, the semantic meaning of the current place?

Home, office, friend's home, transportation location, ... ?



Source
(Internet)

Is the user a male or female? How old is he/she? What types of job is he/she doing? ...

Where is the next place that the user would go to?

Why Mobile Semantic Web ?

- Mobile Data Mining – Here and Now!
- The Performance & Privacy Story
 - **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

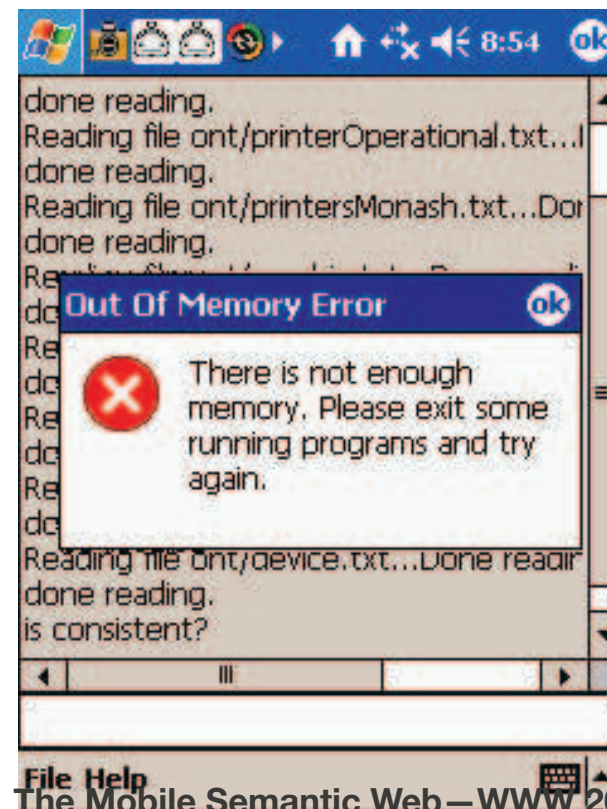
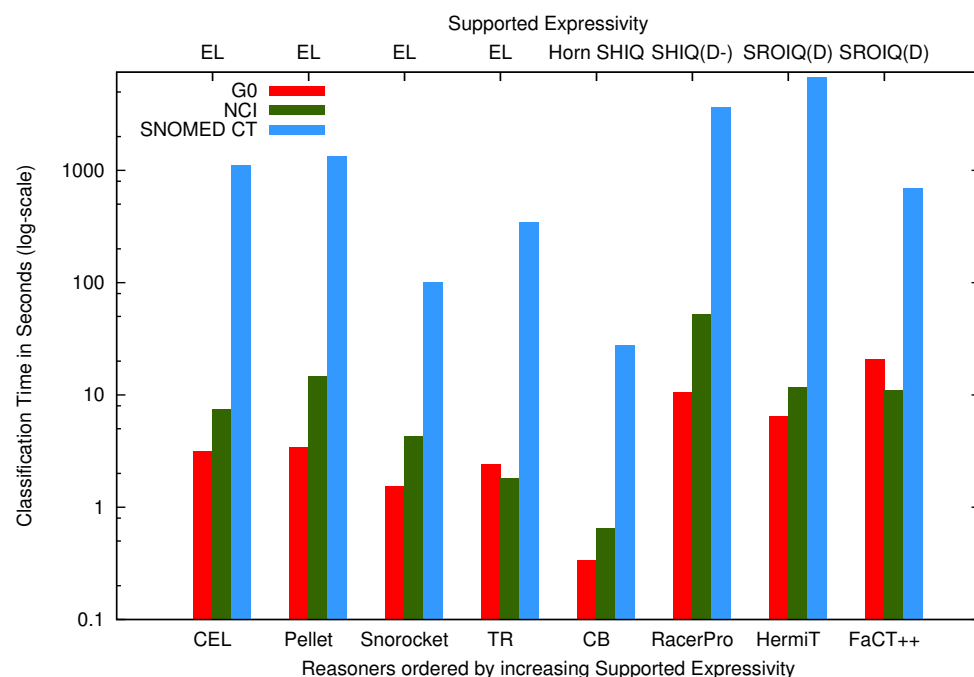
- **Key Challenges**

- 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 quick feedback
 - Attention Span approx 15 s
- Information needs at a high level: *result accuracy*



Mobile Semantic Reasoning

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



© Original Artist
Reproduction rights obtainable from
www.CartoonStock.com

search ID: bgm630

Semantic Reasoning:

What ?

Ontology Reasoning

Basic Description Logics

Reasoning Tasks & Their Complexity

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

Knowledge Base

TBox (schema)

$Animal \sqsubseteq LivingThing$
 $Carnivore \sqsubseteq Animal \sqcap \forall eats.Animal$
...

Rbox (roles)

$eats \sqsubseteq consumes$

ABox (instance)

$simba \in Carnivore$
...



© Disney Inc.

Description Logics: Main Ingredients

- 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, roles & individuals
 - Subsumption, equivalence, disjointness, assertions

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

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

$VegetarianPizza \sqcap NonVegetarianPizza \sqsubseteq \perp$
 $Country(America)$

Description Logics: Syntax & Semantics

- The DL syntax for concept expressions (\mathcal{ALC}):
 - $C ::= \top \mid \perp \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$
 - Other syntaxes available: RDF/XML, etc.
- Set-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 $\Delta^{\mathcal{I}}$

Syntax	Interpretation	Intuition
\top	$\Delta^{\mathcal{I}}$	Everything
\perp	\emptyset	Nothing (empty set)
A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	A class is a set of individuals
$C_1 \sqcap C_2$	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$	Intersection of 2 sets of individuals
$C_1 \sqcup C_2$	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$	Union of 2 sets of individuals
$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	Subtraction from the domain

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, Sattler, Tobies (1999))
 - Nominals & qualified number restrictions
 - OWL 2 DL: $\mathcal{SROIQ}(\mathbf{D})$ (Horrocks, Kutz, Sattler (2006))
 - Complex role inclusions

DL	Example
OWL Lite	$hasTopping \sqsubseteq hasIngredient$
OWL DL	$\geq 3 hasTopping$
OWL 2 DL	$\geq 3 hasTopping.MeatTopping$

A Spectrum of Expressivity (2)

- Less expressive **subsets**
 - Syntactic restrictions on class expressions
 - OWL 2 EL: \mathcal{EL}^{++} (Baader, Brandt, Lutz (2005))
 - Designed for efficient reasoning over large TBoxes
 - Especially biomedical ontologies
 - OWL 2 QL: $DL-Lite_{\mathcal{R}}$ (Calvanese et al (2005))
 - Designed for *very* efficient query answering over large Aboxes
 - Ontology-based data access
 - OWL 2 RL: $DLP \ \& \ pD^*$ (Grosz et al (2003); Horst (2005))
 - Designed for efficient rule-based reasoning

Description Logics: Reasoning Tasks

- 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 $\mathcal{I} \models \varphi$, for every axiom $\varphi \in \mathcal{T}$
- For ABox \mathcal{A}
 - Concept assertions $\mathcal{I} \models C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$
 - Role assertions $\mathcal{I} \models R(a, b)$ 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 axiom $\varphi \in \mathcal{A}$
- Can all be reduced to ABox consistency

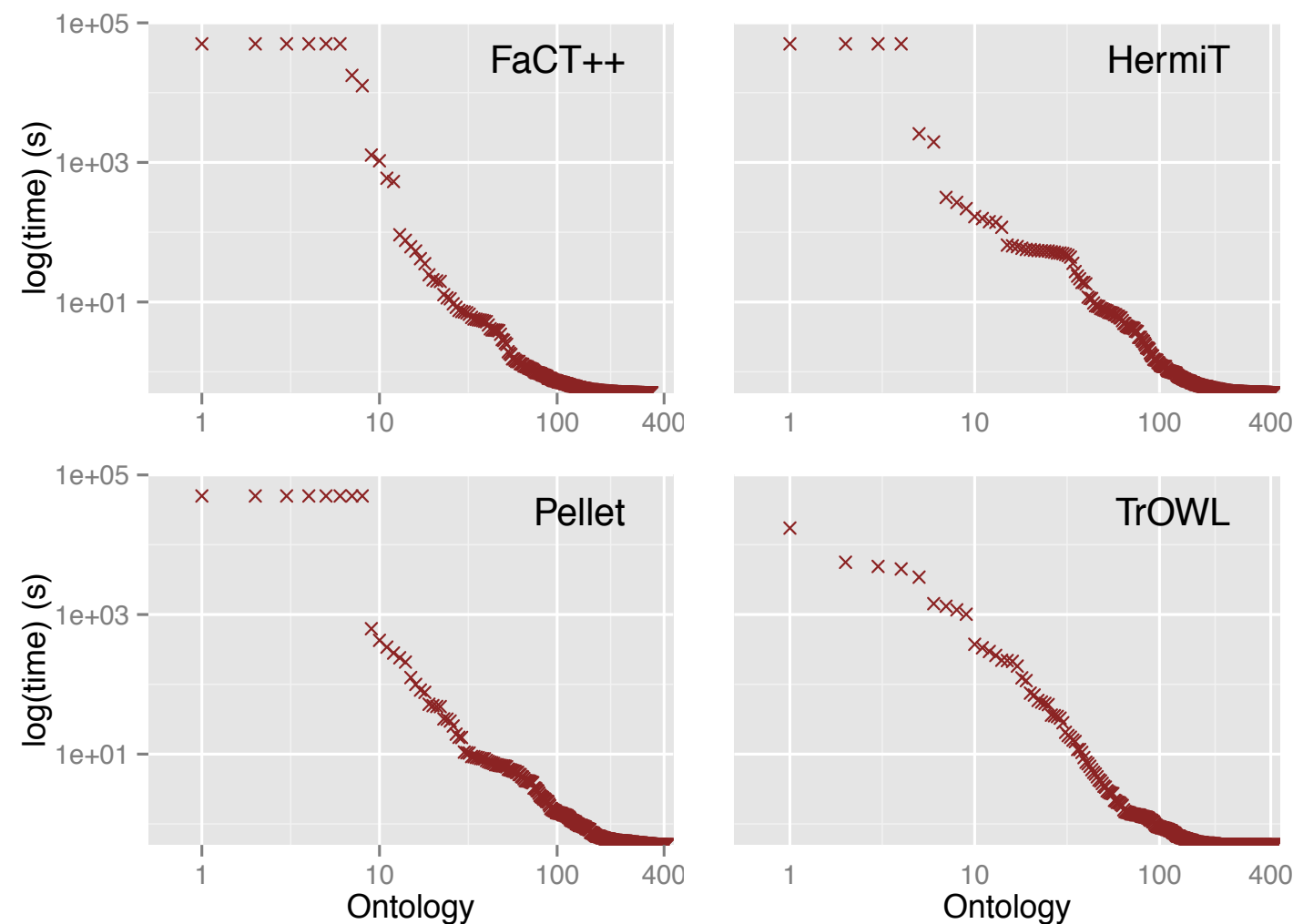
Reasoning Complexity



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
$\mathcal{DL-Lite}_{\mathcal{R}}$ (OWL 2 QL)	NLOGSPACE-complete
OWL 2 RL	PTIME-Complete

Reasoning is **Hard**

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



Lentner, K., Kang, Y., B. et al (2012)

Reasoning is **Hard**

- Difficulty a result of:
 - Ontology:
 - Size (measured in different ways)
 - Language constructs used (union, at most, *etc.*)
 - Reasoner:
 - Tableau algorithms: exhaustive exploration until all ABoxes checked
 - Sound & complete, but **expensive**
 - Optimisation techniques employed

Reasoning Paradigms

- **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))
- **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))

Completion Rules Algorithm

- 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!

Tableau Algorithms

- Foundation of several highly optimised reasoners
 - FaCT++, Racer, Pellet, HermiT, *etc.*
- Support TBox reasoning, *sound & complete*
 - Through reduction to *ABox consistency checking*
 - E.g., 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
- 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
 - $\{C(a), \neg C(a)\} \in \mathcal{A}$
 - $\{\perp(a)\} \in \mathcal{A}$
 - $\{(\leq n \ r)(a)\} \cup \{r(a, b_i) | 1 \leq i \leq n + 1\} \cup \{y_i \neq y_j | 1 \leq i < j \leq n + 1\} \subseteq \mathcal{A}$
for $a, y_1, \dots, y_{n+1} \in N_I, r \in N_R, n \in \mathbb{N}$
 - ...
- Transformation (expansion) rules
 - Can be deterministic or nondeterministic
 - Source of complexity

Expansion Rules

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

\sqcap -rule

$$\begin{array}{l} (C_1 \sqcap C_2)(a) \in \mathcal{A} \\ C_1(a) \notin \mathcal{A} \wedge C_2(a) \notin \mathcal{A} \end{array}$$

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

\sqcup -rule

$$\begin{array}{l} (C_1 \sqcup C_2)(a) \in \mathcal{A} \\ C_1(a) \notin \mathcal{A} \wedge C_2(a) \notin \mathcal{A} \end{array}$$

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

\exists -rule

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

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

\forall -rule

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

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

Tableau Algorithm: An Example

- Individuals $\{x_0, x_1, \dots, x_7\} \subseteq \mathcal{A}$
- 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 $C_0 \equiv \exists R_1.(\geq 1 R_2) \sqcap \exists R_1.(C_1 \sqcap \exists R_1.(C_2 \sqcap C_3))$
- **Negated** $\mathcal{A} \cup \{((\forall R_2.(\leq 0 R_2)) \sqcup (\forall R_1.(\neg C_1 \sqcup \forall R_1.(\neg C_2 \sqcup \neg C_3))))(x_0)\}$ closed?

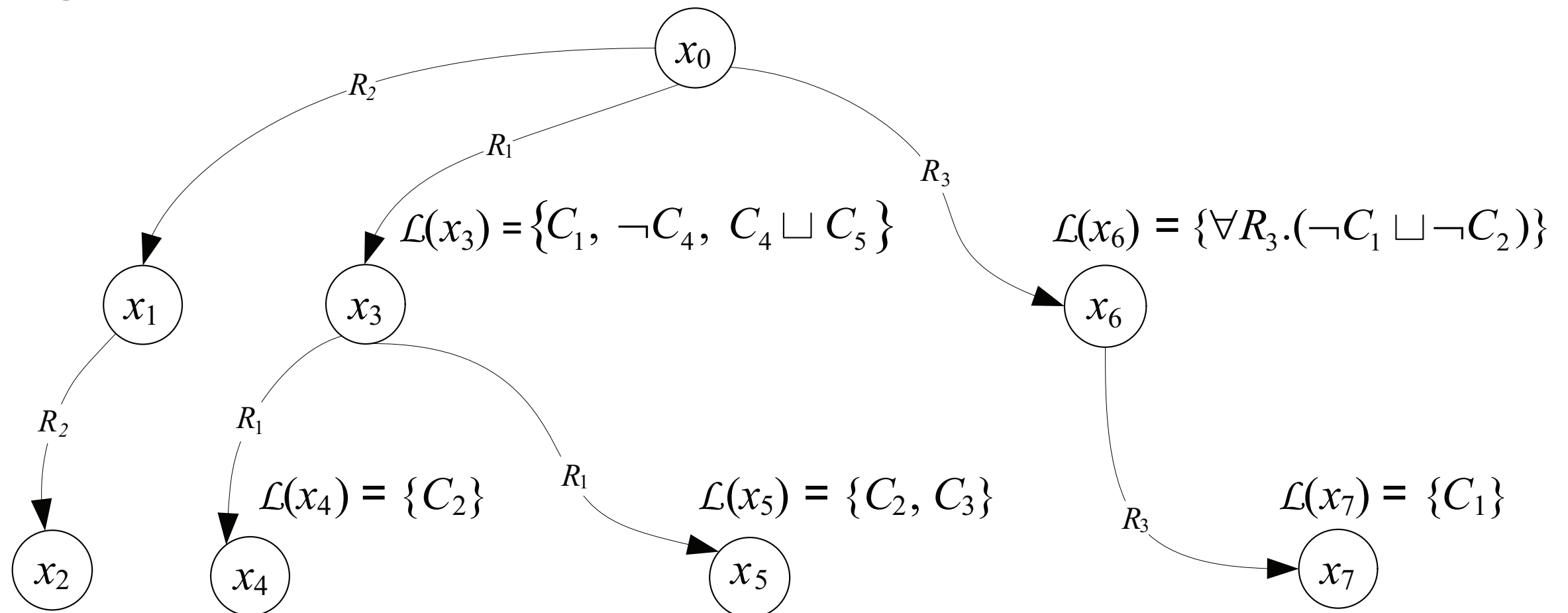


Tableau Algorithm: An Example

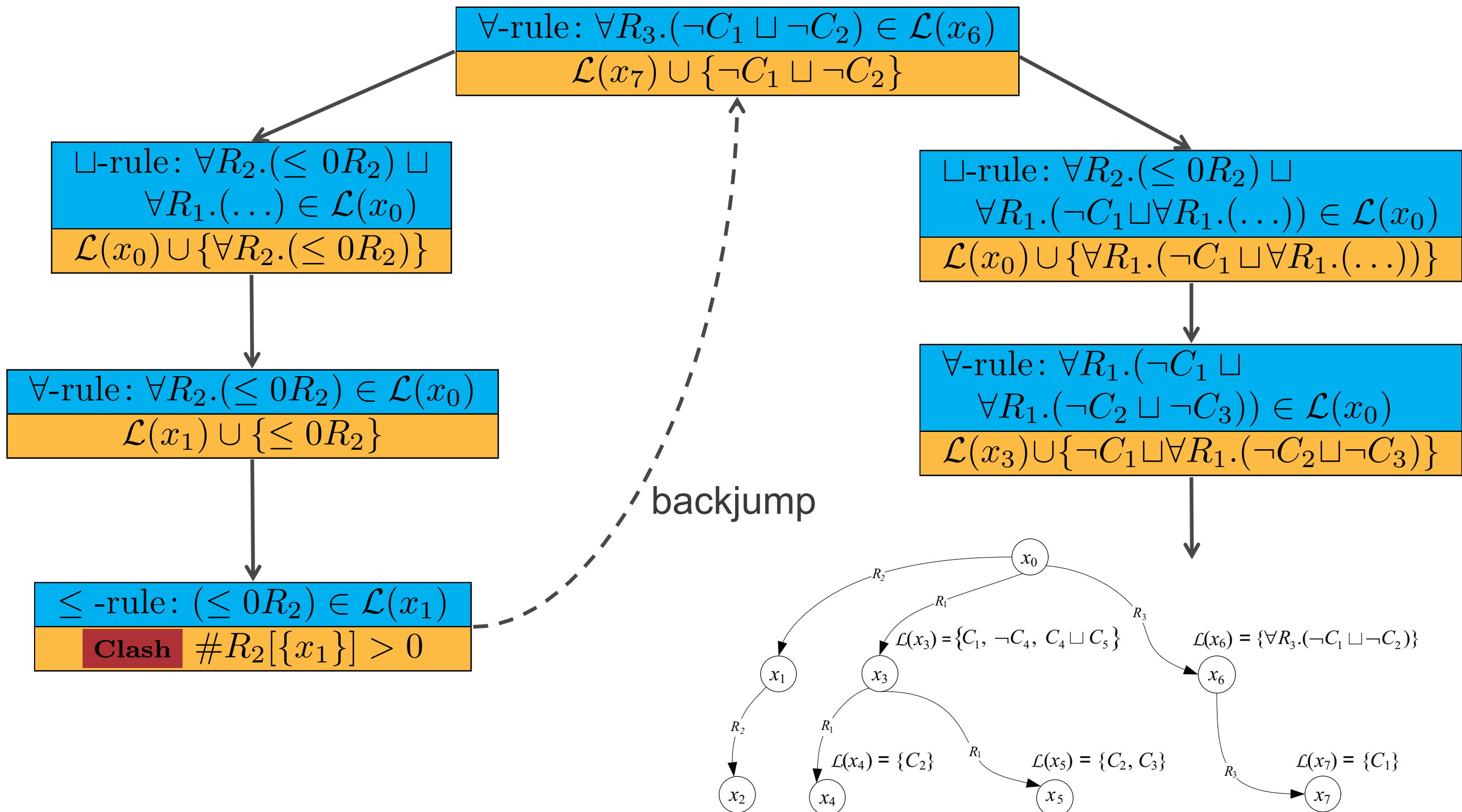


Tableau Algorithm: An Example

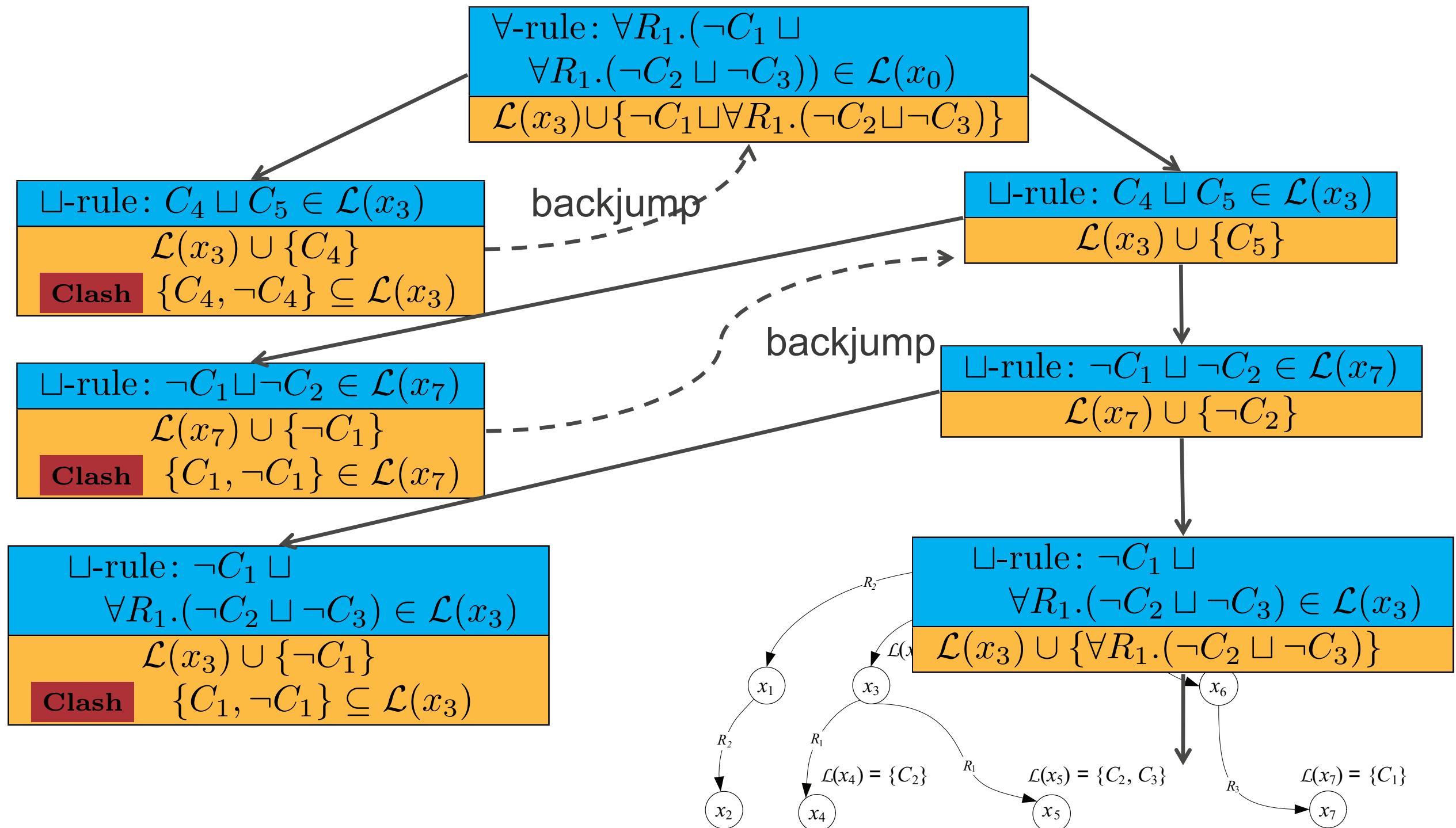
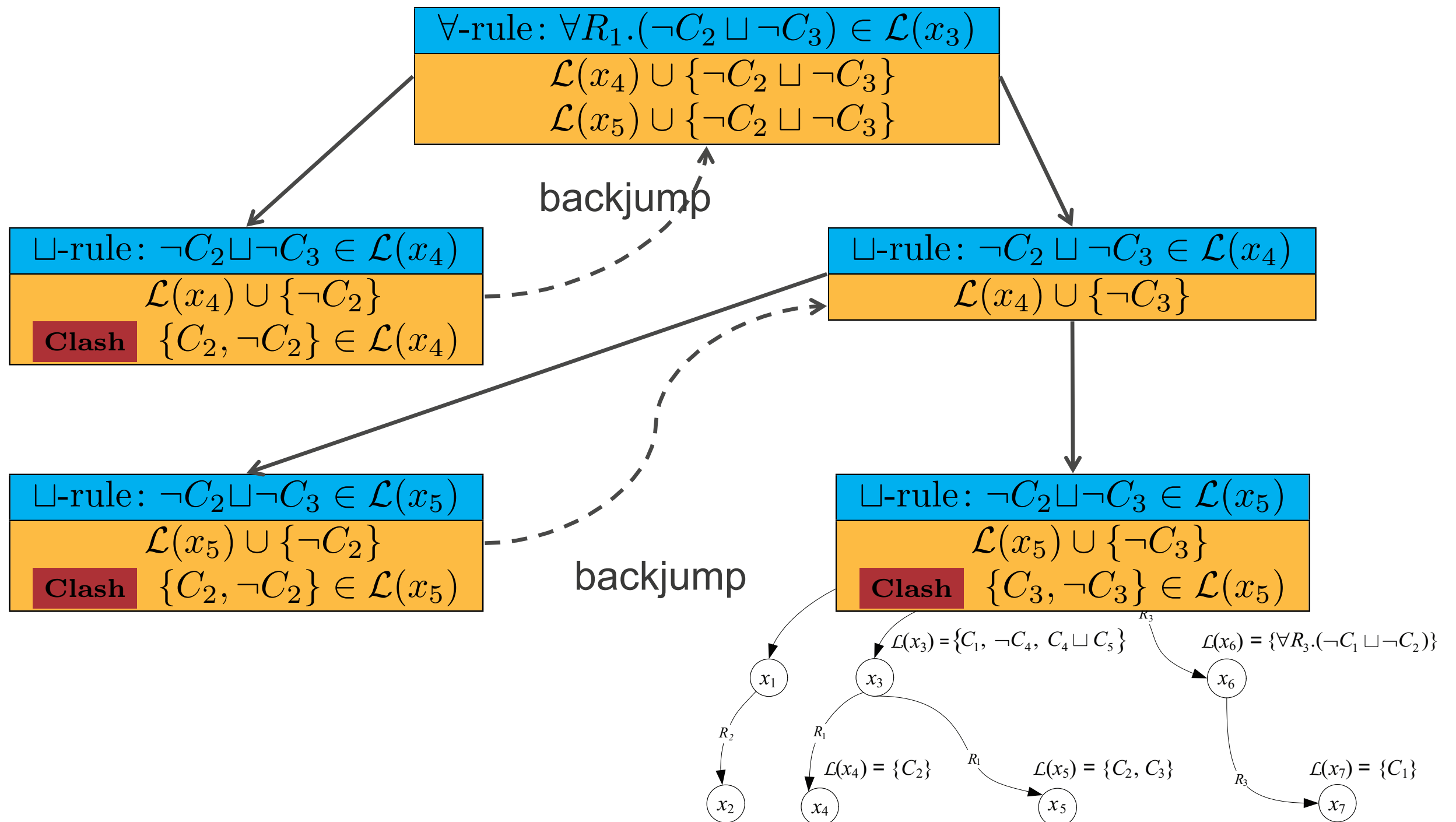


Tableau Algorithm: An Example



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** can improve efficiency
 - Completeness trade-off

Mobile Semantic Reasoning:

How ?

Optimising mobile reasoning: 3 Strategies

- 1. Persistent, adaptive and incremental reasoning**
2. Predictive models for reasoning performance
3. Logical optimisation

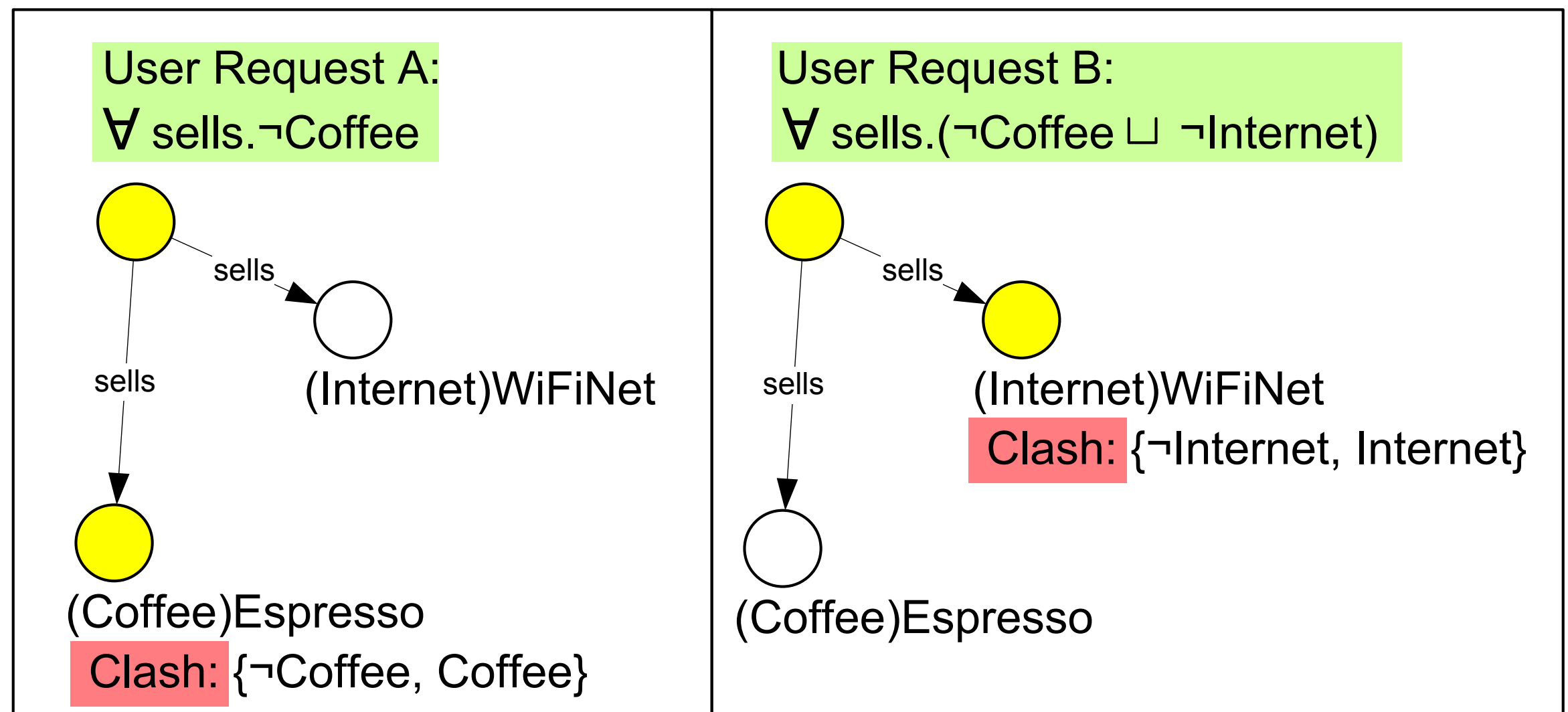
Mobile Semantic Reasoning:

Strategy 1:

Persistent, Adaptive and Incremental Reasoning

Caching Strategy (1)

- Tableaux transformation rules which contribute to a clash (i.e. a positive match) are cached: **mTableaux** (Steller & Krishnaswamy (2008, 2009))



Caching Strategy (2)

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

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

$\mathcal{L}(x_4) \cup \{\neg C_2 \sqcup \neg C_3\}$
--

$\mathcal{L}(x_5) \cup \{\neg C_2 \sqcup \neg C_3\}$
--

- 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 (3)

- 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

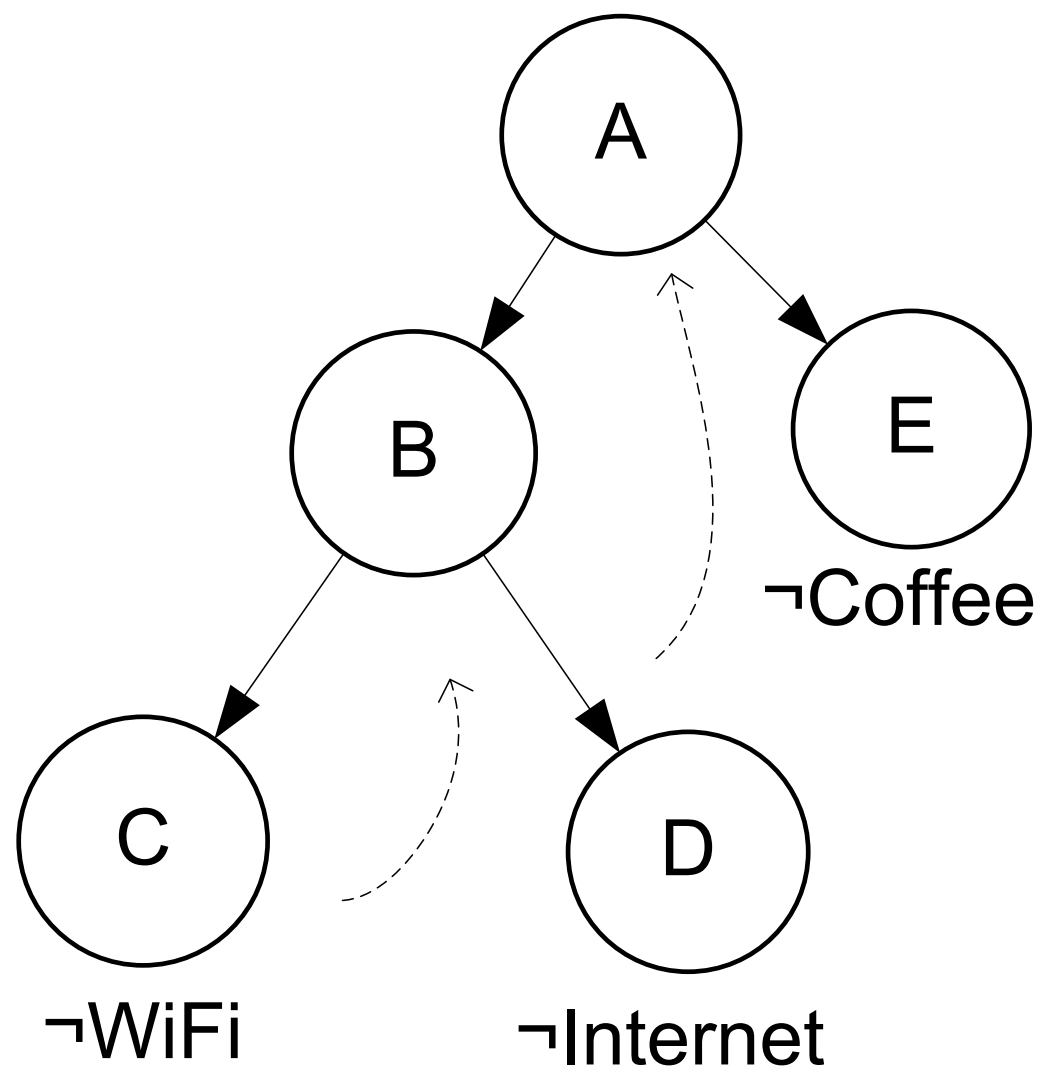
Adaptive Inference

- Typically current reasoners:
 - Depth-first, arbitrary ordering of user requirements
 - Complete entire inference process before a result is provided: “all or nothing” principle
- **mTableaux** adaptive Inference strategy (Steller & Krishnaswamy (2008, 2009)):
 - 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

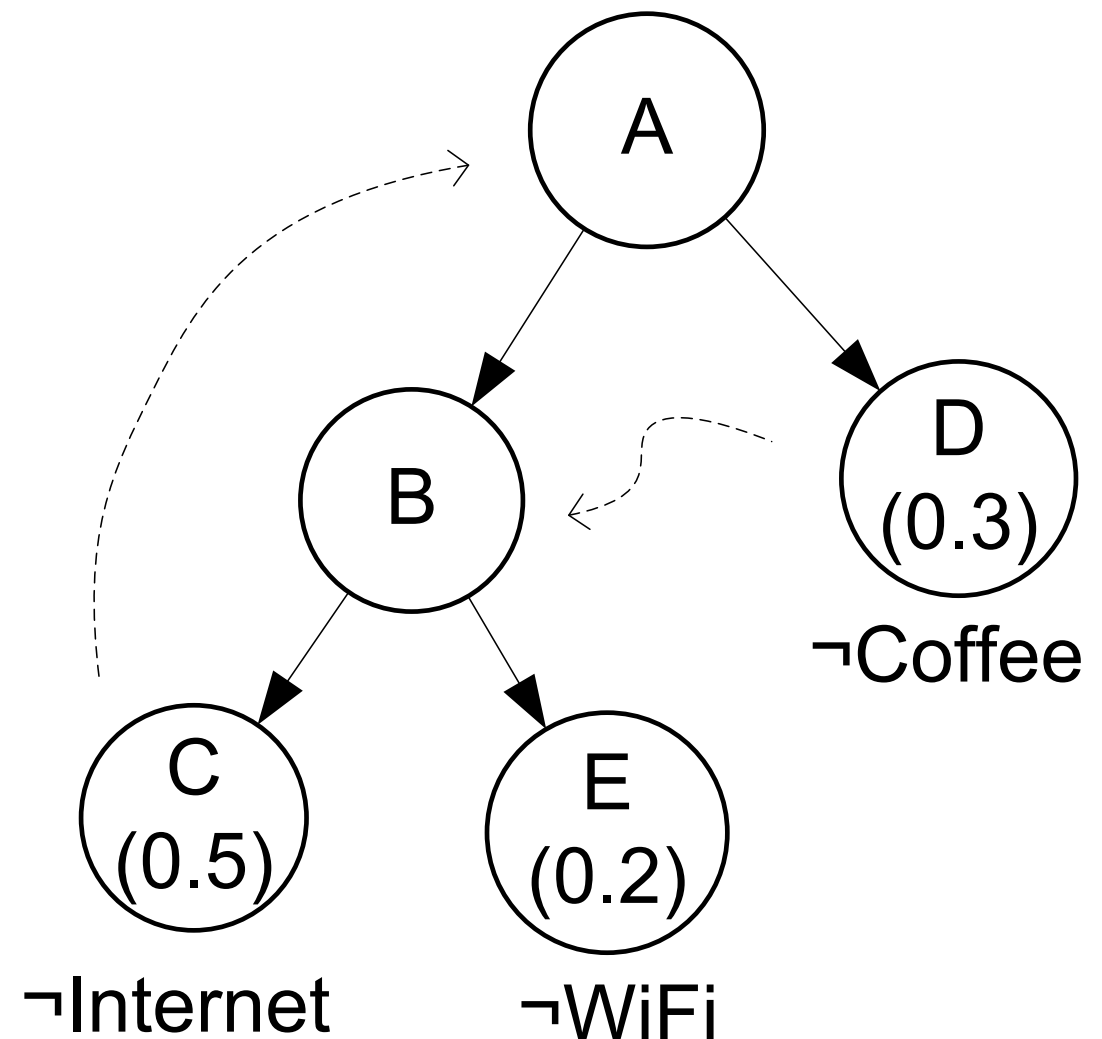
User Request: $((\neg \text{WiFi} \sqcup \neg \text{Internet}) \sqcup \neg \text{Coffee})$

Standard Tableaux:



Adaptive Inference:

Order: $\neg \text{Internet}$, $\neg \text{Coffee}$, $\neg \text{WiFi}$

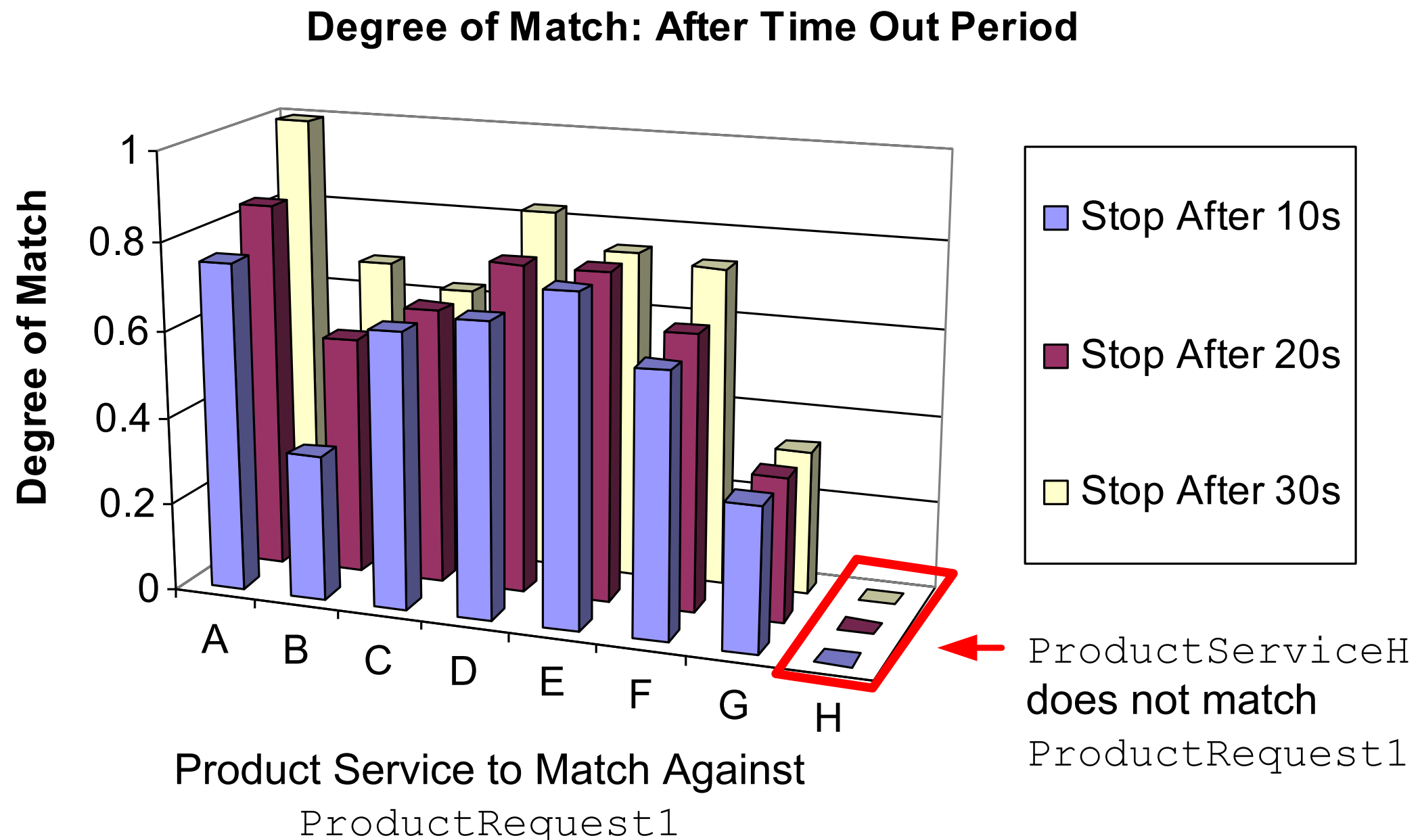


Adaptive Inference - Considerations

- Priority expansion of disjunction elements
 - Queuing and expansion ordering
- Simultaneously open / unfinished branches
 - Branch identifiers
 - State management
- Degree of match computation

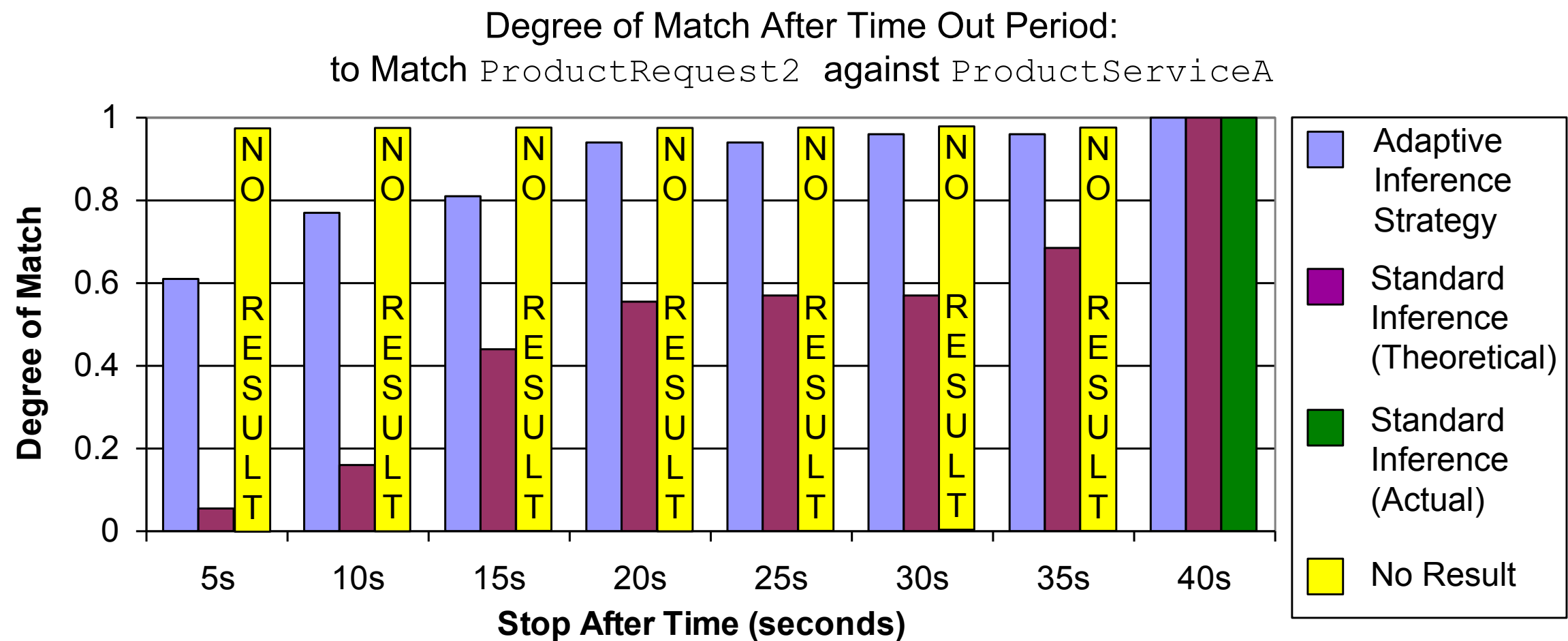
Adaptive Inference Strategy

Degree of Match – Product Scenario (after timeout)



Adaptive Inference Strategy

Degree of Match (after timeout) - Product Scenario



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

Optimising mobile reasoning: 3 Strategies

1. Persistent, adaptive and incremental reasoning
- 2. Predictive models for reasoning performance**
3. Logical optimisation



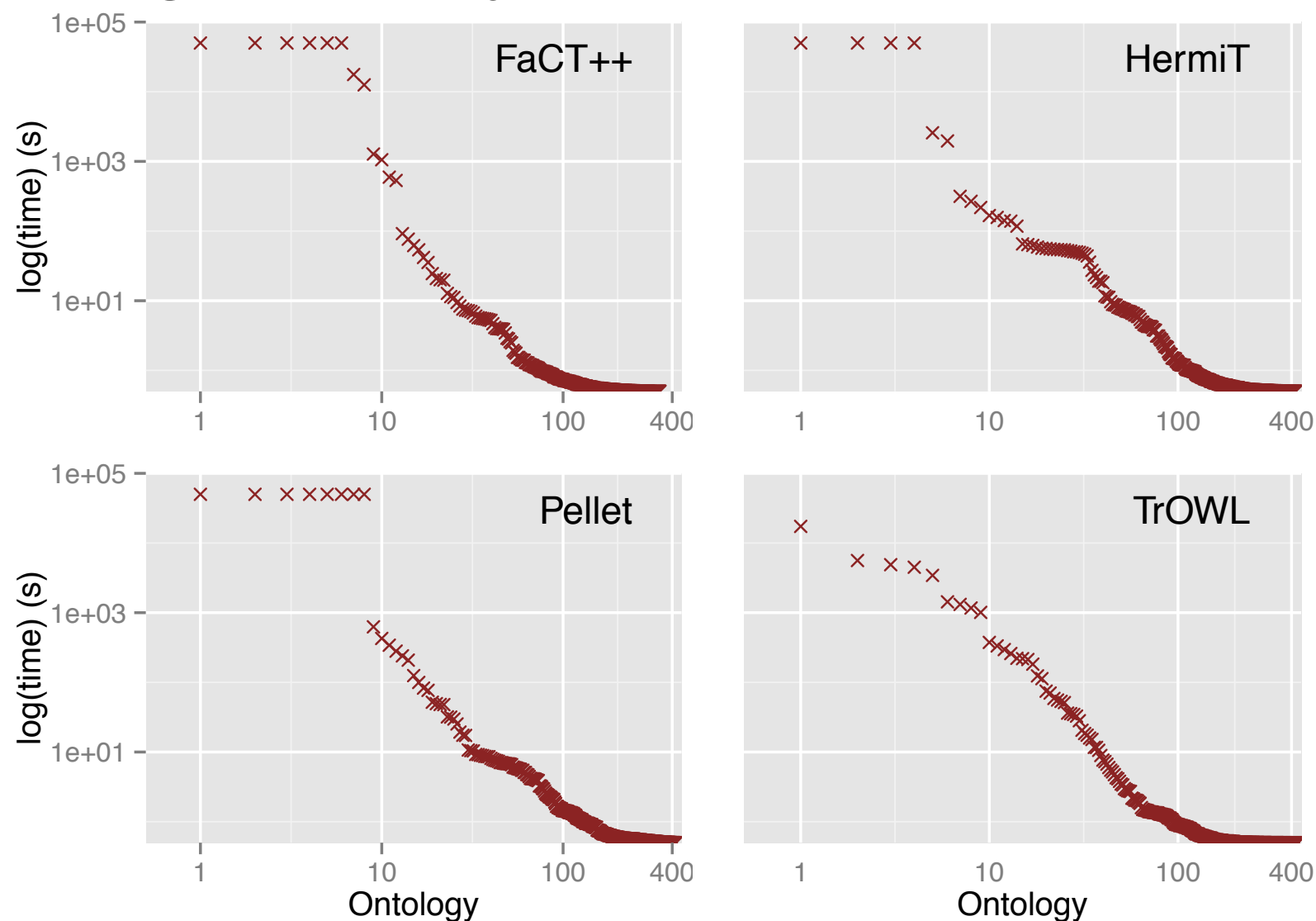
Mobile Semantic Reasoning:

Strategy 2:

Predictive Models for Reasoning Performance

A Need to Understand Reasoning Performance

- Recall, reasoning can be *very hard*



Kang, Y.-B. et al (2012)

A Need to Predict Reasoning Performance

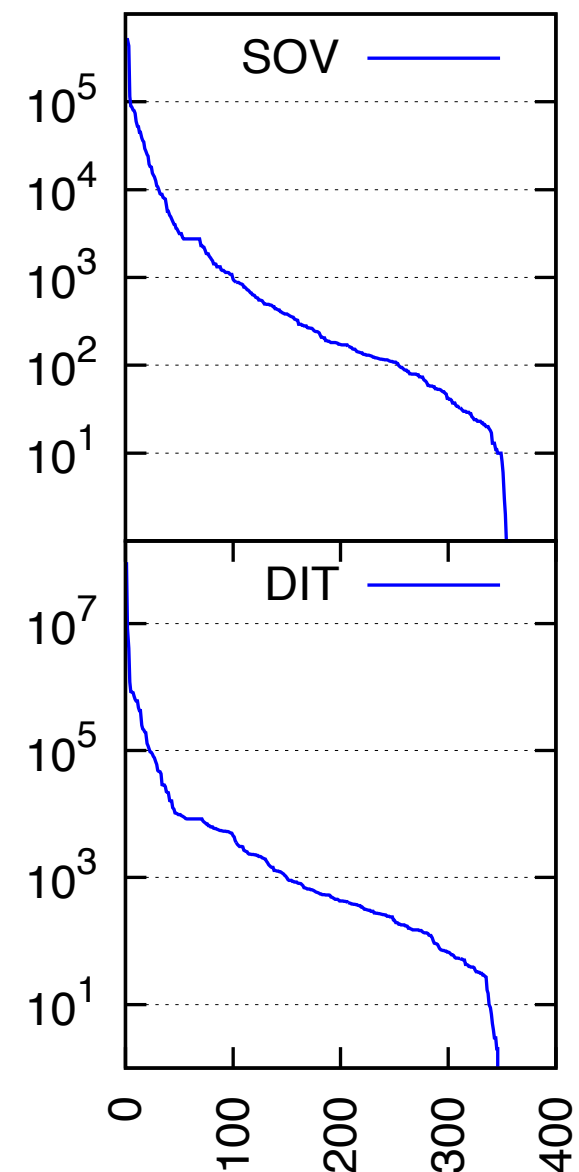
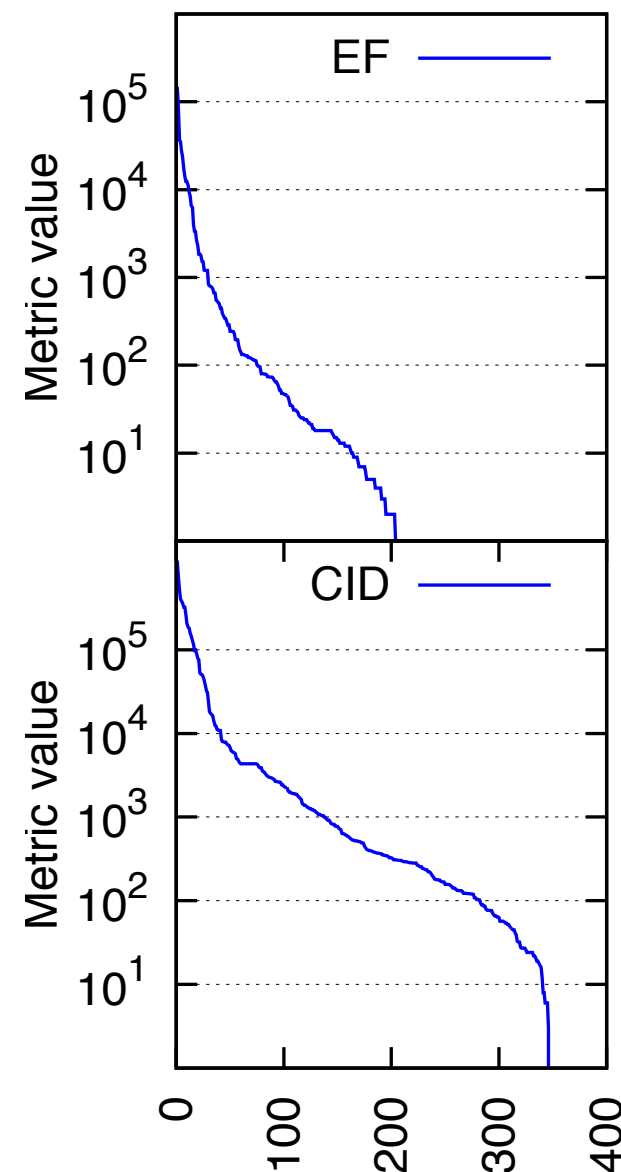
- Ontologies are different
 - Size doesn't tell the whole story
 - Interactions of expressions/axioms important source of complexity
- Reasoners are different
 - Different reasoning paradigms & optimisation techniques
 - Drastic performance differences on the same ontology
- Hence,
 - What to do when facing design choices?
 - What to do when efficiency is a hard constraint?
 - How long will a reasoner take?

Performance Prediction: Take 1

- Prediction models for ontology reasoner performance (Kang et al 2012)
 - Given an (*ontology*, *reasoner*) pair, predict the performance of the *reasoner* on the *ontology*
 - Predict *discretised* reasoning time: *classification*
- Use *ontology metrics* as features:
 - Rationale: ontologies may be difficult for different reasons
 - Metrics capture different aspects of complexity
- Train *prediction models* on subsets of features
 - Not every feature is equally useful
- Identify *key metrics* that impact performance the most

Ontology Metrics

- 27 metrics organised into 4 categories
- ONT: ontology-level metrics
 - Overall size & complexity
- CLS: class-level metrics
 - Complexity about named classes
- ACE: anonymous class expressions
 - Complexity of different types of expressions
- PRO: property expressions & axioms
 - Complexity related to properties



Prediction Model Construction (1)

1. Data collection

- 350+ ontologies of various sizes & difficulty levels
- 4 reasoners
- Collect metric values for all ontologies
- Collect reasoning performance for all (ontology, reasoner) pairs

2. Reasoning time discretisation

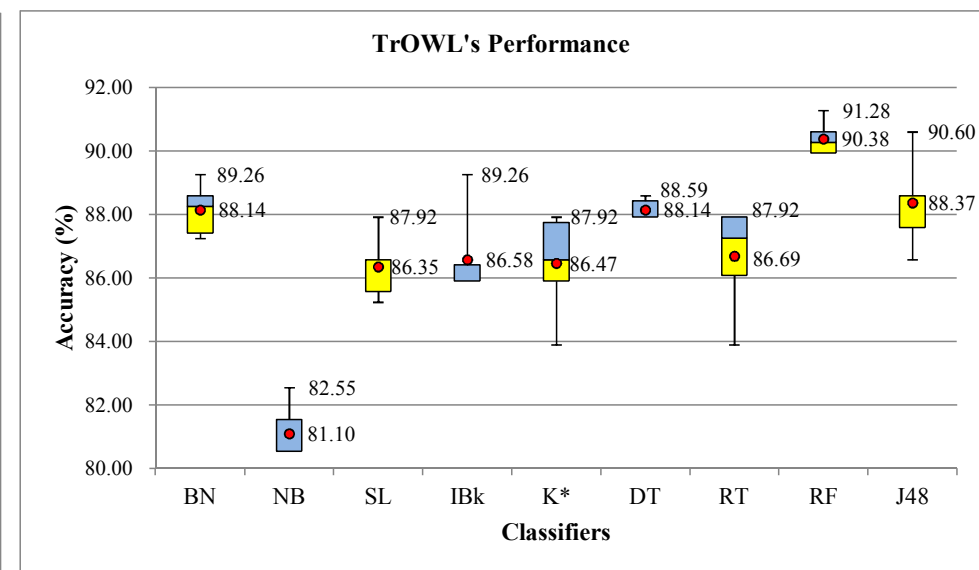
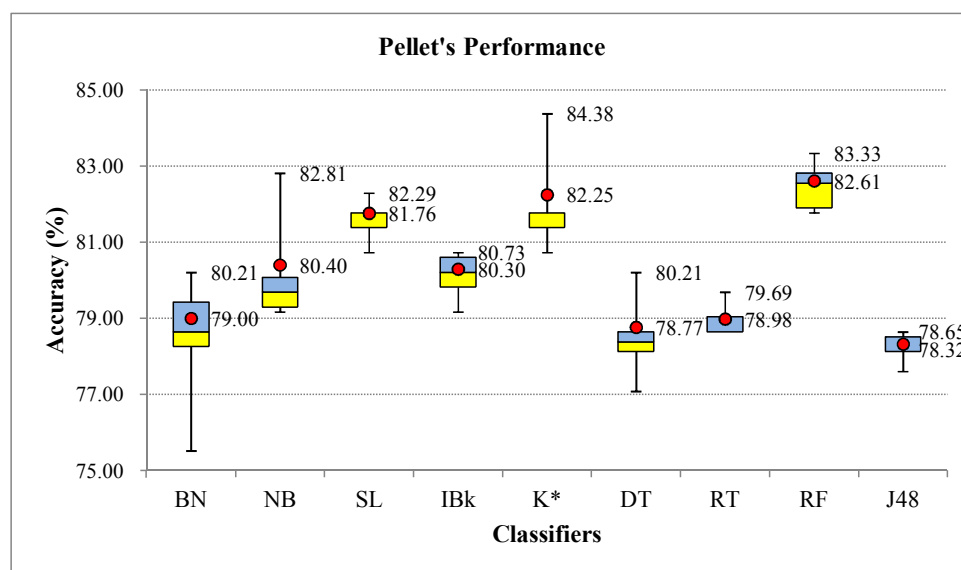
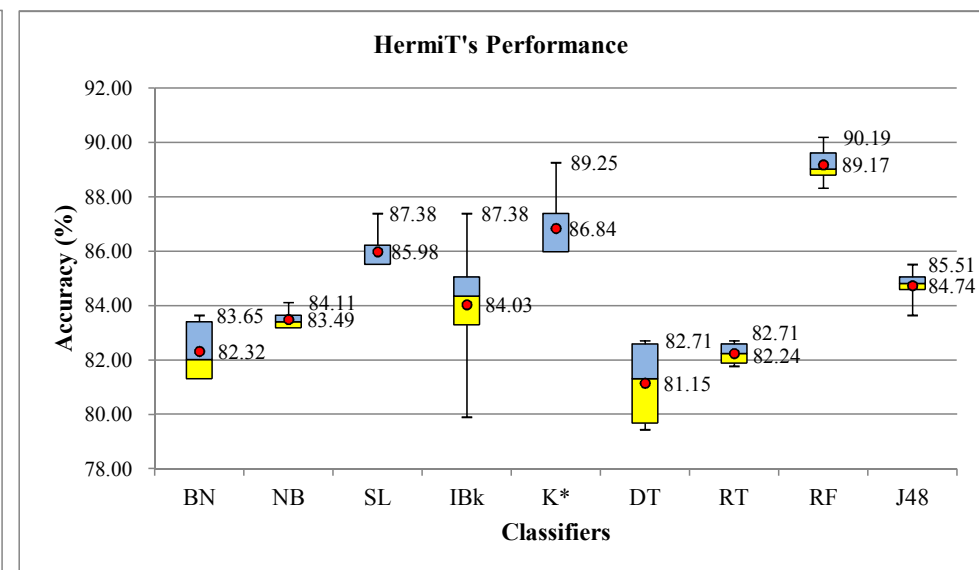
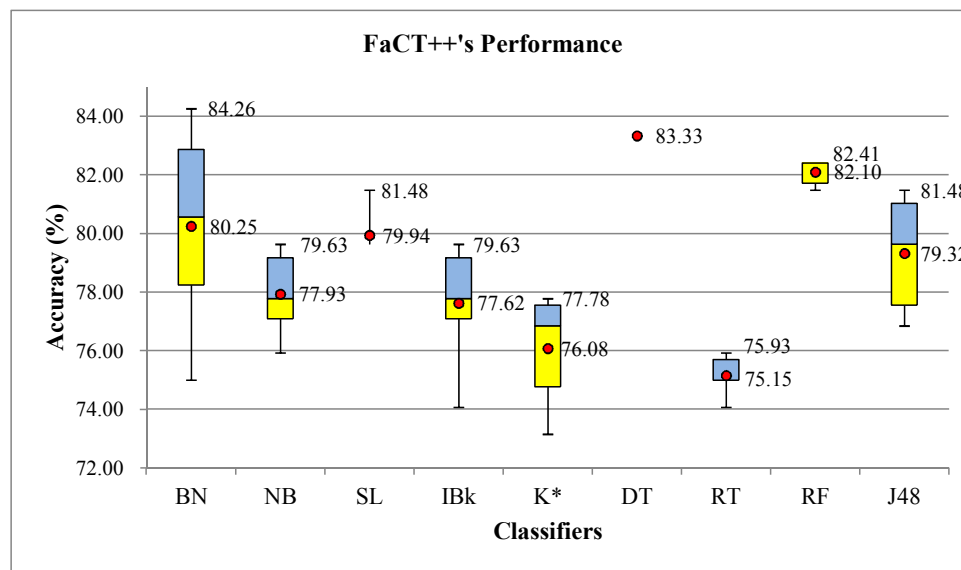
Label	Classification time	FaCT++	HermiT	Pellet	TrOWL
A	$0.01s < A \leq 1s$	75	154	126	105
B	$1s < B \leq 10s$	16	35	38	17
C	$10s < C \leq 100s$	6	12	12	13
D	$100s < D$	11	13	16	14
Total		269	291	330	337

Prediction Model Construction (2)

1. Feature selection
 - Not all features are equally useful
 - Apply 6 ranking-based feature selection algorithms
2. Model training
 - Not all models are equally accurate
 - Train many models, not only one: 9 models (Bayesian, decision tree, rules, etc.)
 - Prediction performance measure by *accuracy*
 - 10-fold cross validation to evaluate prediction performance

Prediction Model Evaluation (1)

- Model accuracy

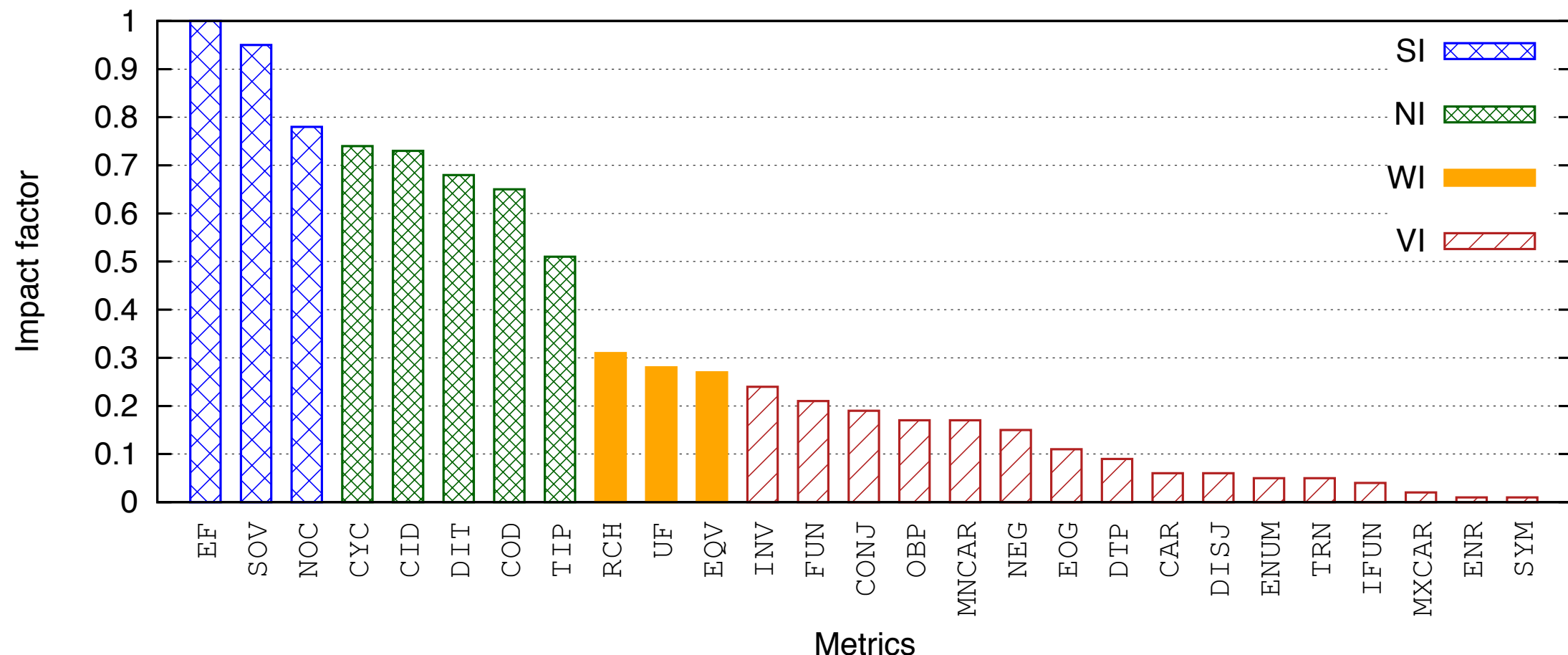


Prediction Model Evaluation (2)

- Average prediction accuracy for all models exceed 80%
- Metrics make good features for reasoning performance prediction – worst accuracy is still close to 80% (Naïve Bayes)
- Random forest the most accurate
- Random forest the most stable

Key Metrics Identification

- What makes reasoning hard?
- Identify key metrics that impact reasoning performance the most
- Calculation: combining metric frequency & weight in prediction models



Performance Prediction: Take 2

- Prediction models for ontology reasoner performance
 - Instead *discretised* reasoning time, prediction *actual* reasoning time
 - *Regression analysis*
- Use *ontology metrics* as features:
 - Same 4 categories: ONT, CLS, ACE & PRO
 - Significantly expanded metrics set: 91 metrics
 - Capture complexity more comprehensively
- Training: a regression model for each reasoner
- Application: identify performance hotspots

Prediction Model Construction

1. Data collection
 - 450+ ontologies
 - 6 OWL reasoners: FaCT++, HermiT, JFact, Pellet & TrOWL
2. Data preprocessing
 - Cleansing, normalisation, metrics removal, splitting
3. Regression model building
 - Training a random forest model for each reasoner

Prediction Model Evaluation

- Prediction performance metrics:
 - Coefficient of determination R^2 : the higher the better
 - Root mean square error RMSE: the lower the better
- The models are highly accurate

Model	Training set		Test set	
	R^2	RMSE	R^2	RMSE
FaCT++	0.853	1.24	0.905	0.810
HermiT	0.868	1.13	0.913	0.918
JFact	0.834	1.37	0.922	0.786
MORe	0.882	0.81	0.833	0.986
Pellet	0.835	1.15	0.904	0.913
TrOWL	0.942	0.89	0.934	0.909

Application: Performance Hotspot Identification

- Hotspots (Gonçalves, Parsia, and Sattler 2012) represent performance bottlenecks
 - A small subset of axioms (generated from a *signature*) that significantly impact performance, i.e., for a subset M of ontology O , M is a hotspot if $\#M \ll \#O$ and $RT(O \setminus M, R) \ll RT(O, R)$
 - $\#$ represents the size, RT represents reasoning time on ontology O by reasoner R
 - Their identification provides insights for ontology design & maintenance
- The current identification algorithm (Gonçalves, Parsia, and Sattler 2012) requires exhaustive concept satisfiability checking: can be *expensive*

Regression-based Hotspot Identification: Algorithm

- We apply the *regression model* to this problem
- Given ontology O , reasoner R , number of candidates k
 1. For each concept C in ontology O
 1. *Adaptively* generate candidate M_C from C
 2. Predict reasoning time t_{M_C} of M_C using the prediction model for R
 2. Rank all candidates C by t_{M_C} in descending order
 3. Return the top k candidates

Regression-based Hotspot Identification: Dataset

- Dataset: 8 large bio-ontologies known to contain hotspots (Gonçalves, Parsia, and Sattler 2012)

Ontology	# axioms	# classes	Reasoner	$RT(O, R)$
ChEBI	60,085	28,869	Pellet	174.1
EFO	7,493	4,143	HermiT	78.8
IMGT	1,122	112	HermiT	122.73
			MORe	108.71
			Pellet	> 3,600
VO	8,488	3,530	HermiT	65.21
			Pellet	> 3,600
			TrOWL	115.31
NCIt	116,587	83,722	HermiT	1,020.9
			MORe	643.2

Regression-based Hotspot Identification: Results

- Evaluation criteria: no. of hotspots identified & no. of tests required

Ontology	Reasoner	# hotspots	# tests	% avg # hotspot	% avg RT boost
ChEBI	Pellet	10	10	0.5%	96%
EFO	HermiT	10	16	1.2%	75.5%
IMGT	HermiT	3	14	6.8%	79.4%
	MORe	10	16	8.7%	76.6%
	Pellet	2	14	6.8%	> 99.98%
VO	HermiT	10	10	4.6%	98.5%
	Pellet	10	10	4.5%	> 98.1%
	TrOWL	10	10	4.5%	97.8%
NCIt	HermiT	10	10	3.3%	90.8%
	MORe	10	10	2.5%	88.7%

Regression-based Hotspot Identification: Observations

- Hotspot identification
 - Our method identifies more hotspots using fewer tests for most (O , R) pairs than (Gonçalves, Parsia, and Sattler 2012)
 - Generating candidates is fast (polynomial to $\# O$)
 - Making prediction is really fast
 - Most pairs are identified with very few tests (less than 20)
- Reasoning performance
 - Different reasoners may have dramatically different performance on the same ontology
 - Reinforces the need for understanding & predicting performance

Optimising mobile reasoning: 3 Strategies

1. Persistent, adaptive and incremental reasoning
2. Predictive models for reasoning performance
- 3. Logical optimisation**

Mobile Semantic Reasoning:

Strategy 3:

Logic-level Optimisation and Tuning

KRHyper

- KRHyper: Kleemann and Sinner (2006)
 - Novel Tableaux reasoner for First Order Logic (FOL) for deployment on resource constrained devices
 - 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.
- Performance
 - Better performance than RacerPro for small test cases, not performing as well for larger tests.
 - 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

Mini-ME

- Mini-ME: Ruta et al. (2008a,b,2012)
 - 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
 - Mini-ME: an *ALCN* reasoner Android (Ruta et al. 2012)
 - Improved memory footprint compared to previous version

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

	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

Optimisation Techniques in mTableaux: At a Glance

- mTableaux: a framework for light-weight mobile DL inference (Steller & Krishnaswamy (2008, 2009))
 - 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 strategies
 - Selective application of transformation rules (**ST**)
 - Selective application of the disjunction rule (**SD**)
- Caveat: realisation/consistency checking not performed
 - Ontology assumed to be consistent
 - Hence no completeness guarantee
 - False negatives possible, false positives not

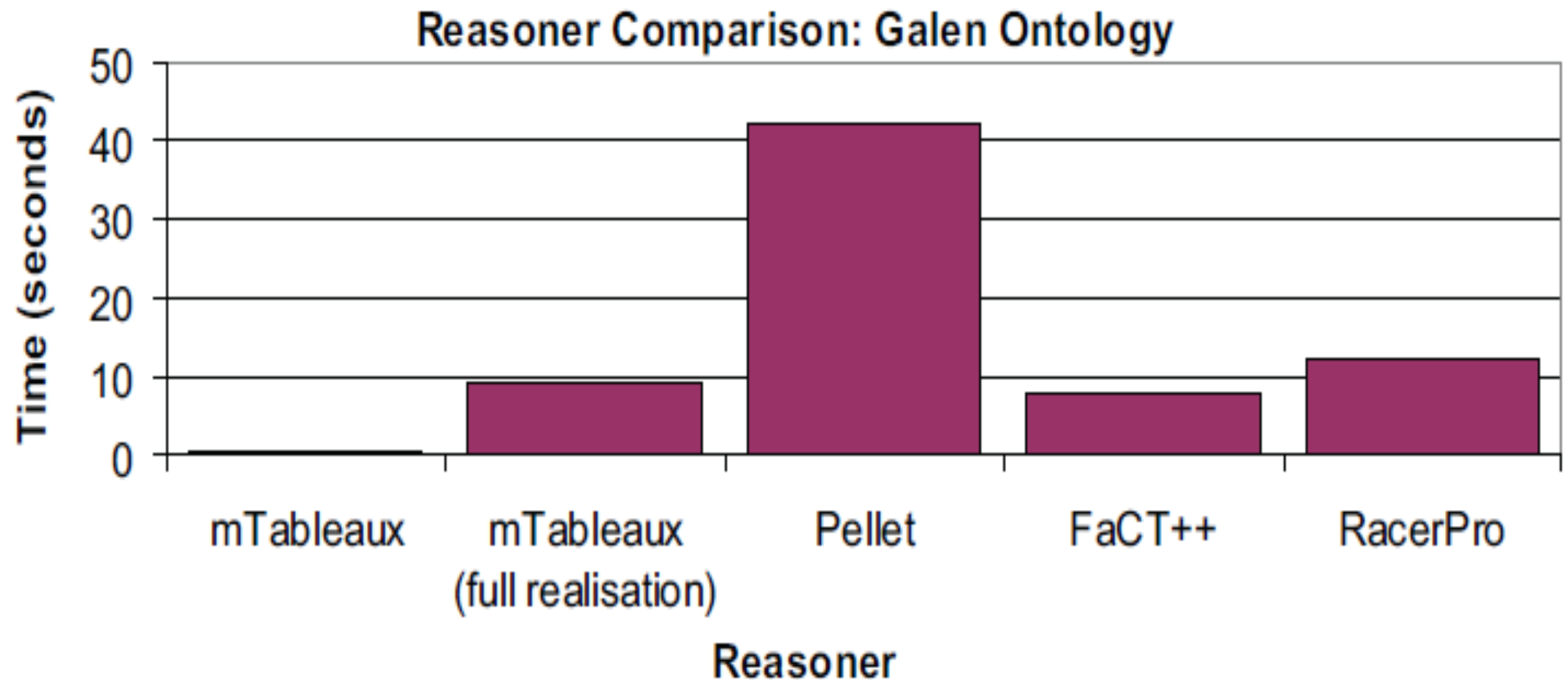
Selective Transformation Rule Application (ST)

- Tableaux transformation (expansion) rules are applied to only a subset of individuals, which relate to the service description being checked
 - E.g., a user searches for an Internet café that sells both Internet and coffee $\exists \textit{sells.Coffee} \sqcap \exists \textit{sells.Internet}$
 - That a venue *hasPlayground* is not of interest
- More formally, given an ABox A , tableaux transformation (expansion) rules are only applied to a subset of individuals ST include
 - the individual of interest x ,
 - iteratively, other individuals that are connected to individuals in ST through properties in *universally qualified* class expressions, and their super properties, and
 - nothing else.

Selective Disjunction Rule Application (SD)

- The disjunction rule (\sqcup -rule) is non-deterministic, and expensive
 - It generates new ABoxes, all of which need to be explored
 - Hence increases search space
- Aim of SD: reduce the no. of \sqcup -rule applications
- Tableaux disjunction expansion rule is only applied to disjunctions which contain class concepts which relate to the user request
 - Hence reducing search space
- E.g., looking for Internet café? $\exists \text{ sells. Internet } \sqcap \exists \text{ sells. Coffee}$
 - Apply \sqcup -rule to $\text{Tea } \sqcup \text{ Coffee}$, not to $\text{Pizza } \sqcup \text{ VideoGame}$

Evaluation of mTableaux Optimisation: Comparison with Other Reasoners



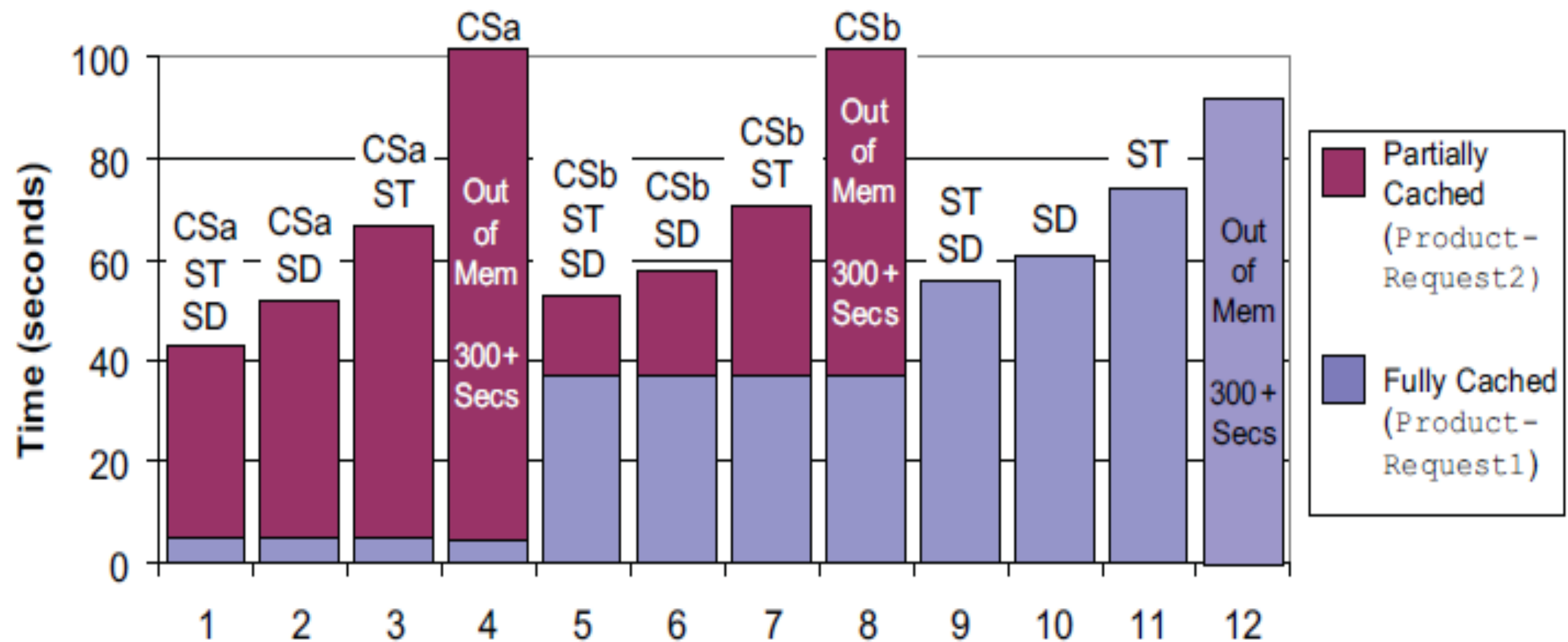
Evaluation of mTableaux Optimisation: Combinations of Techniques

- Evaluated on resource-constrained mobile devices

Technique	1	2	3	4	5	6	7	8	9	10	11	12
ST	X		X		X		X		X		X	
SD	X	X			X	X			X	X		
Caching (CSa)	X	X	X	X								
Caching (CSb)					X	X	X	X				

Evaluation

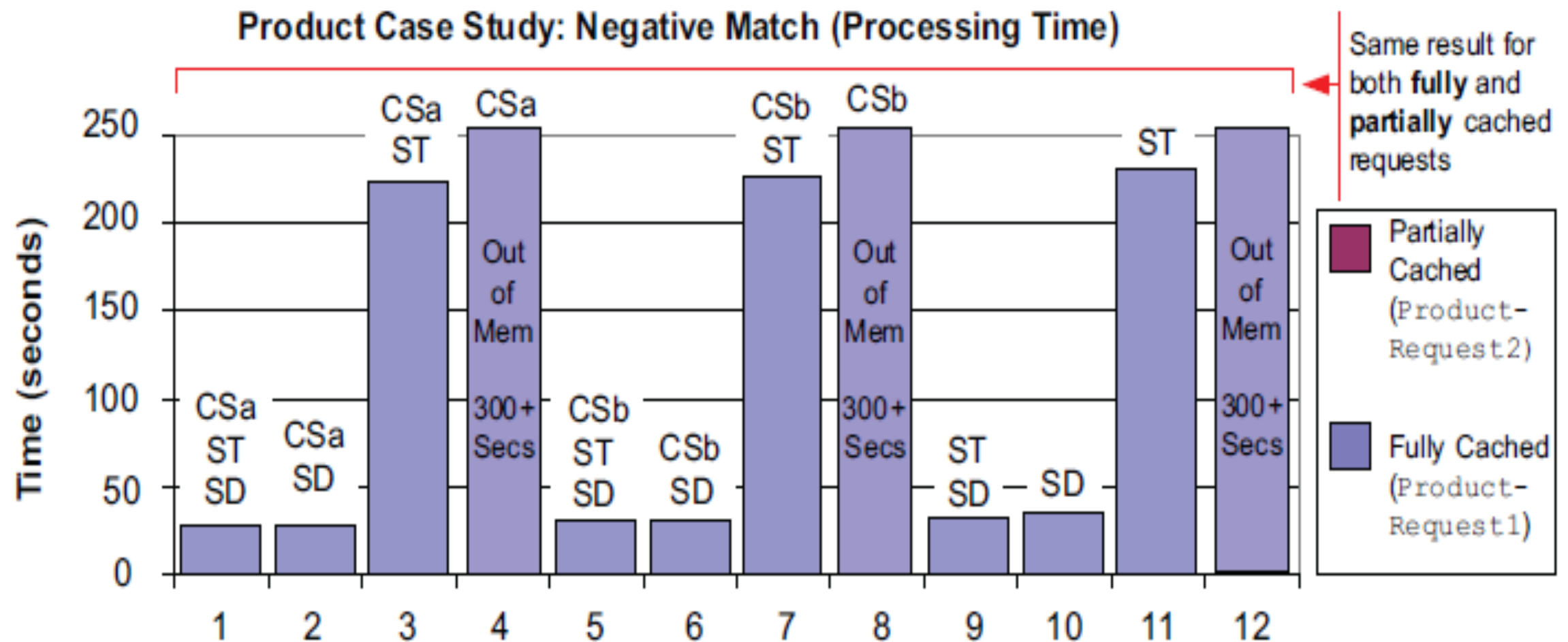
Product Case Study: Positive Match (Processing Time)



Test: Combination of mTableaux strategies enabled

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

Evaluation



Test: Combination of mTableaux strategies enabled

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

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.
- 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.
- The combination of adaptive, caching and logical optimisation techniques provides the fastest / most efficient response-time

Related Work & Recent Development

ELK on Android

- Direct porting of ELK on Android 4.2: Kazakov & Klinov (2013)
 - ELK 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

Ontology	Workers	Google Nexus 4				PC			
		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 [◇]	0.57	0.97
	Android1	4.90	0 [◇]	14.88	33.22
	Android2	3.42	0 [◇]	10.43	20.77
Wine	PC	10.39	0 [◇]	6.54	2.22
	Android1	2196.05	0 [◇]	511.97	194.12
	Android2	1609.32	0 [◇]	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!

Mobile Semantic Reasoning:

What's Next ?

Future Directions

- Meta-reasoning
 - Given an ontology, find a reasoner that will be the most efficient
 - Making use of prediction models: work in progress
- Hybrid reasoning – mobile + cloud
 - Determine the most appropriate platform for a reasoning job
 - Task decomposition
- Comparative evaluation of reasoners and reasoning strategies for resource utilisation
 - Adaptation to other constraints – resources, confidence
 - Understand and optimise for reasoning impact on CPU, RAM, power consumption



Thank you!

Questions?

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.
- 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.

References

- 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.
- 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).
- 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.
- Chen, H., Finin, T. and Joshi, A. (2004). Semantic web in the context broker architecture, *2nd International Conference on Pervasive Computer and Communications (PerCom 04)*, IEEE, Orlando, Florida, pp. 277 – 286.
- 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

- Gonçalves, R. S.; Parsia, B.; and Sattler, U. (2012). Performance heterogeneity and approximate reasoning in description logic ontologies. In *ISWC 2012*, 82–98. Springer Berlin Heidelberg.
- Grimm, S., Watzke, M., Hubauer, T., & Cescolini, F. (2012). Embedded EL+ Reasoning on Programmable Logic Controllers. *ISWC 2012* (pp. 66-81). Springer Berlin Heidelberg.
- Grosz, 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 framework 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.

References

- Horrocks, I., & Patel-Schneider, P. F. (2003). Reducing OWL entailment to description logic satisfiability. *ISWC 2003* (pp. 17-29). Springer Berlin Heidelberg.
- Horrocks, I.; Sattler, U.; and Tobies, S. (1999). Practical Reasoning for Expressive Description Logics. In *LPAR-6*, 161–180. Springer
- 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 mobile phones, *INAP'06*, Springer-Verlag, Fukuoka, Japan, pp. 135 – 147.

References

- 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.
- 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.
- 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.
- 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.
- Ren, Y., Pan, J. Z., & Zhao, Y. (2010). Soundness Preserving Approximation for TBox Reasoning. *AAAI* (pp. 351-356).

References

- 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 *AAAI-22* (pp. 1093-1098). AAAI Press.
- Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *JWS*, 5(2), 51-53.

References

- Srinivasan, N., Paolucci, M. and Sycara, K. (2005). Semantic web service discovery in the OWL-S IDE, *39th International Conference on System Sciences*, Vol. 6, IEEE, Hawaii, USA, pp. 109 – 119.
- 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.
- 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 matchmaking among heterogeneous software agents in cyberspace, *Autonomous Agents and Multi-Agent Systems, Kluwer Academic* 5(2): 173 – 203.

References

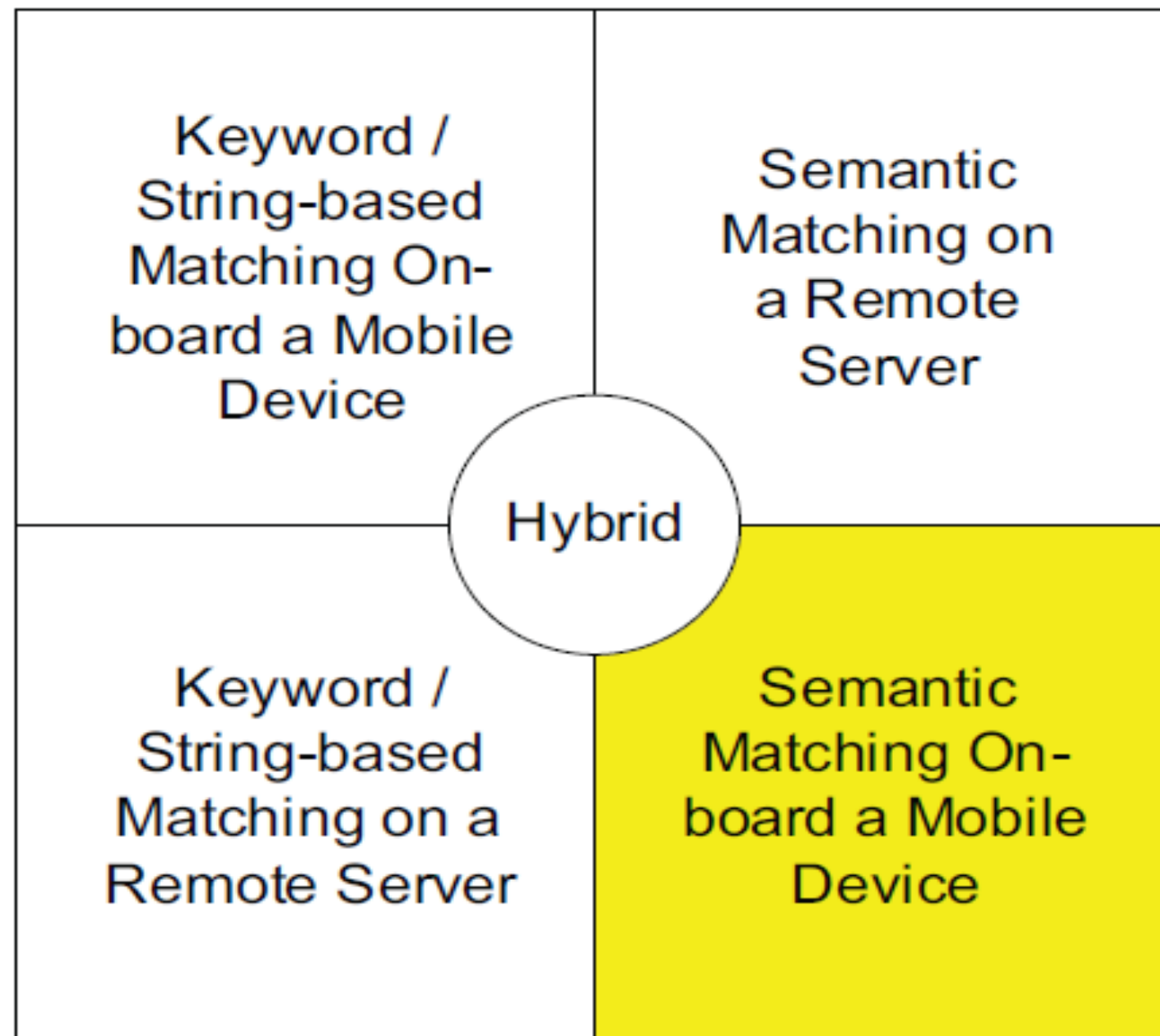
- ter Horst, H. J. (2005). Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *Journal of Web Semantics*, 3(2), 79-115.
- Tsarkov, D., & Horrocks, I. (2006). FaCT++ Description Logic Reasoner: System Description. *Automated reasoning*, 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).



For your reading:

Service Matching in Mobile Environments

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 than 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 – Support 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 – Support 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 – Support 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 – Support 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
- The 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
 - Current open source / commercial reasoners such as Pellet, KAON2, FaCT++, RacerPro considerably resource intensive
 - Cannot function on small resource constrained devices