Graph Rewriting for Model Construction in Modal Logic

Bilal SAID

Institut de Recherche en Informatique de Toulouse - IRIT
Université de Toulouse

29 January 2010

Locating my work

	Automated reasoning	
Theory	x	Practice
	Graph rewriting	

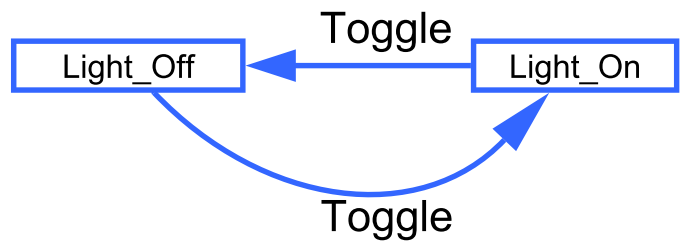
Panorama

1. Modal logics
2. Model Construction
3. Event-driven pattern matching

Outline

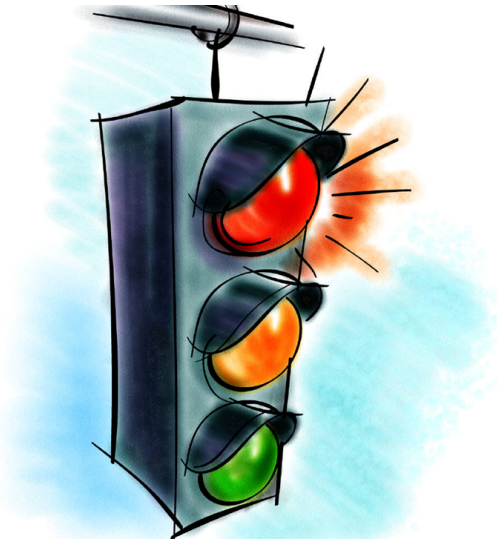
Example: Switch-Bulb system

Example: Switch-Bulb system



Various systems, situations...

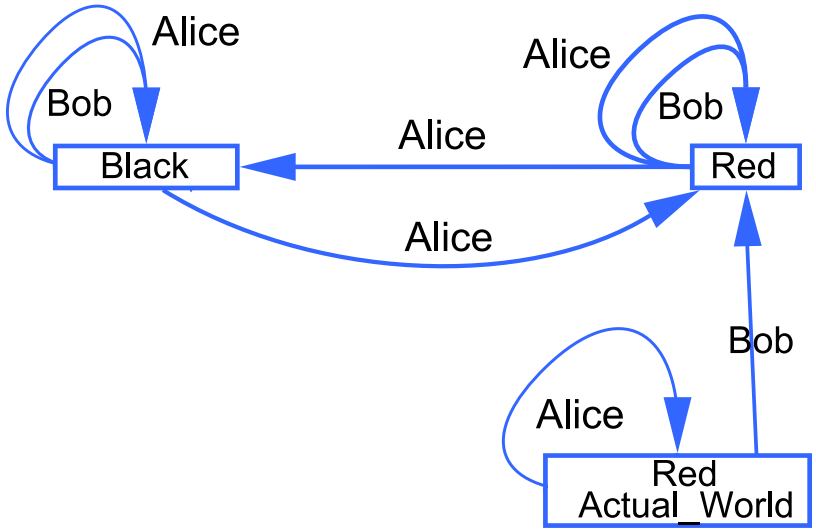
Guess-the-card game



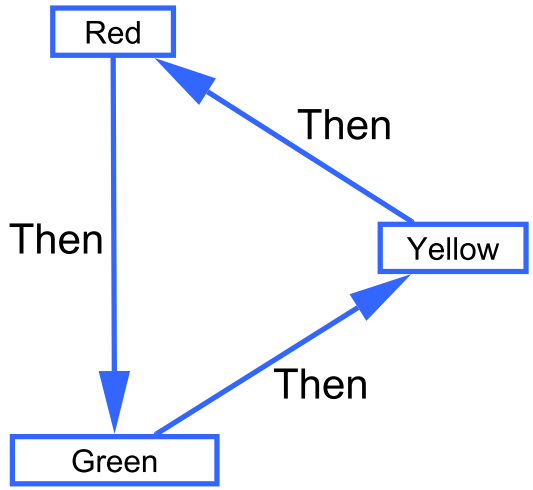
Traffic-light

...

Various systems, situations...



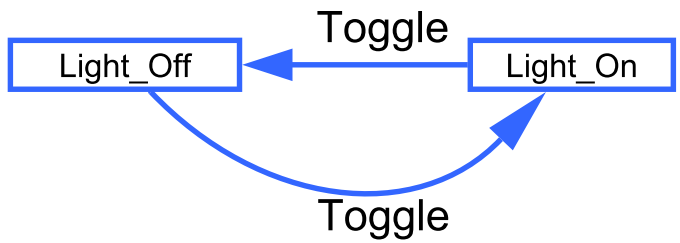
Knowledge



Time

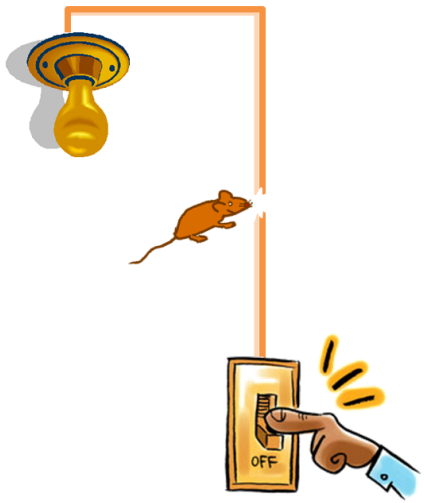
...

Switch-Bulb system

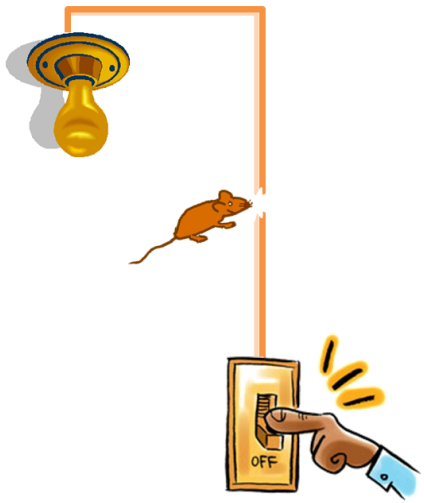


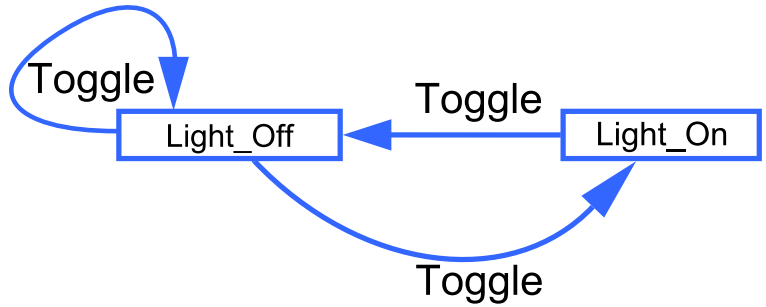
at the state "Light_Off", after "Toggle" necessarily "Light_On"

Switch-Bulb-Mouse

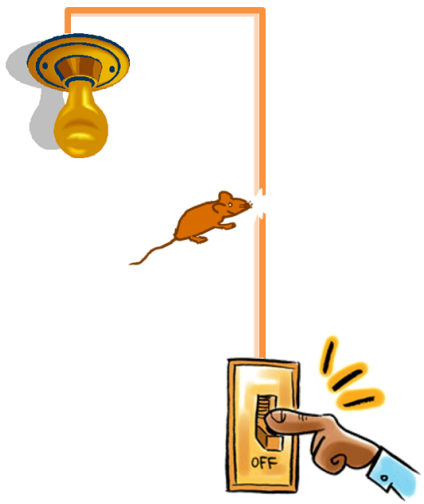


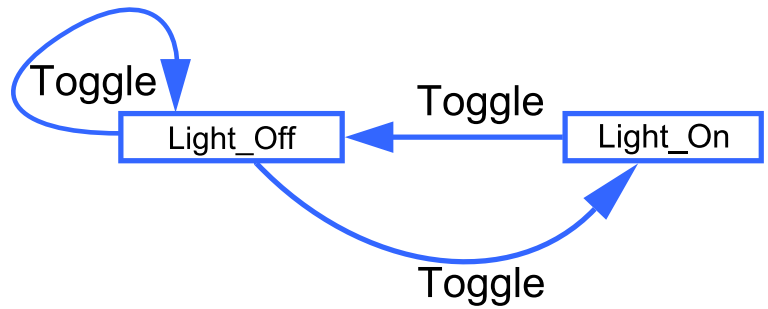
Switch-Bulb-Mouse





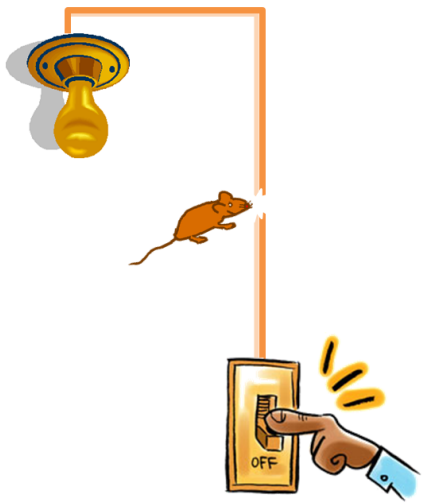
Switch-Bulb-Mouse

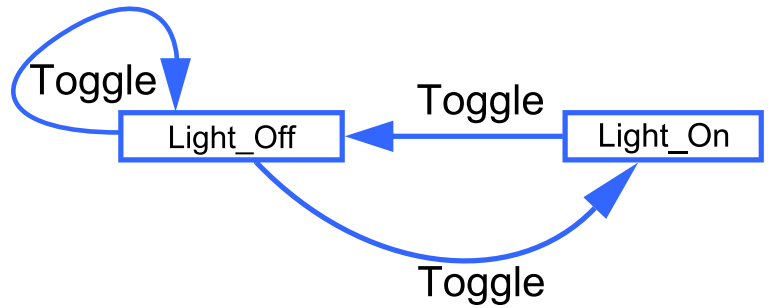




at state "Light_Off", after "Toggle" not necessarily "Light_On"

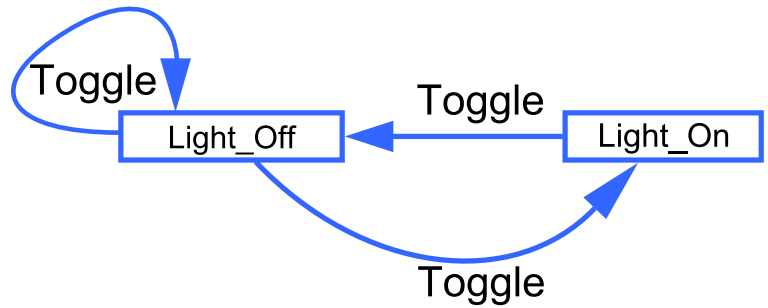
Switch-Bulb-Mouse





at state "Light_Off", after "Toggle" not necessarily "Light_On"
i.e. possibly not "Light_On"

Switch-Bulb-Mouse



at state "Light_Off", after "Toggle" not necessarily "Light_On"
i.e. possibly not "Light_On"
i.e. possibly "Light_Off"

Formal language: Formulas

- "Light_Off" and after "Toggle" necessarily "Light_On"
  $Light\_Off \quad \wedge \quad [Toggle] \quad Light\_On$
- "Light_Off" and after "Toggle" possibly "Light_Off"
  $Light\_Off \quad \wedge \quad \langle Toggle \rangle \quad Light\_Off$

Generic sentences:

Necessarily A $\quad \Box A$
Possibly A $\quad \Diamond A$

How to evaluate these formulas?

Formal language: Formulas

- "Light_Off" and after "Toggle" necessarily "Light_On"
  $Light\_Off \quad \wedge \quad [Toggle] \quad Light\_On$
- "Light_Off" and after "Toggle" possibly "Light_Off"
  $Light\_Off \quad \wedge \quad \langle Toggle \rangle \quad Light\_Off$

Generic sentences:

Necessarily A	$\Box A$
Possibly A	$\Diamond A$

How to evaluate these formulas?

Formal language: Formulas

- "Light_Off" and after "Toggle" necessarily "Light_On"
  $Light\_Off \quad \wedge \quad [Toggle] \quad Light\_On$
- "Light_Off" and after "Toggle" possibly "Light_Off"
  $Light\_Off \quad \wedge \quad \langle Toggle \rangle \quad Light\_Off$

Generic sentences:

Necessarily A $\quad \Box A$

Possibly A $\quad \Diamond A$

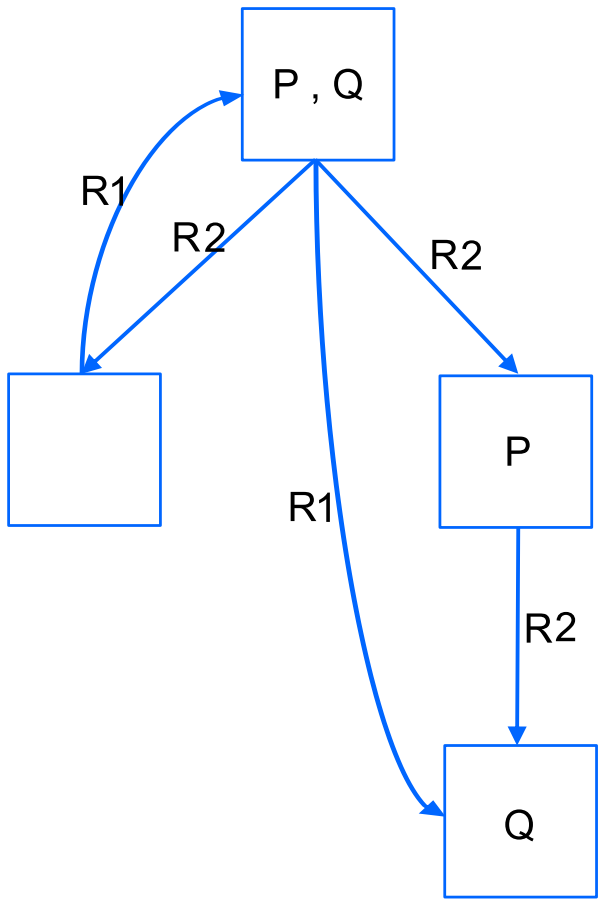
How to evaluate these formulas?

Kripke models

Model
= transition system

- possible Worlds
  = states
- accessibility Relation
  = transitions
- Valuation
  = labeling function

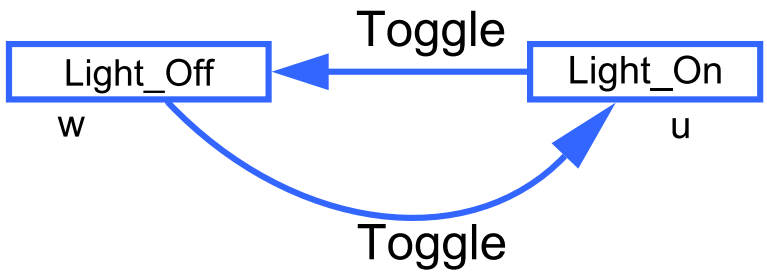
$$M = (W, R, V)$$



Semantics: Truth conditions

- Atoms
  - $M, w \Vdash P$ iff $P \in V(w)$
- Classical operators
  - $M, w \Vdash A \wedge B$ iff $M, w \Vdash A$ and $M, w \Vdash B$
  - $M, w \Vdash A \vee B$ iff $M, w \Vdash A$ or $M, w \Vdash B$
  - ...
- Modal operators
  - $M, w \Vdash \Box A$ iff for all u, if $\underline{wRu}$ then $M, w \Vdash A$
  - $M, w \Vdash \Diamond A$ iff exists u s.t. $\underline{wRu}$ and $M, u \Vdash A$

Example



- $M, u \Vdash \mathit{Light\_On}$
- $M, w \Vdash \mathit{Light\_Off} \wedge [\mathit{Toggle}]\mathit{Light\_On}$
- ...

Various systems, situations...

Examples

- The card is red, Alice knows it and knows that Bob does not
  $Card\_Red \wedge K_{Alice} Card\_Red \wedge K_{Alice} \neg K_{Bob} Card\_Red$
- Always, if it is Red then next it turns out Green
  $\mathrm{G}(Red \rightarrow \mathrm{X} Green)$
- ...

How to take these into account?

- Change truth conditions
- Constraints on R
- Constraints on V

Various systems, situations...

Examples

- The card is red, Alice knows it and knows that Bob does not
  $Card\_Red \wedge K_{Alice} Card\_Red \wedge K_{Alice} \neg K_{Bob} Card\_Red$
- Always, if it is Red then next it turns out Green
  $\mathrm{G}(Red \rightarrow \mathrm{X}\, Green)$
- ...

How to take these into account?

- Change truth conditions
- Constraints on R
- Constraints on V

Constraints on R

One relation:

- Transitive
  future of future is future
- Reflexive
  I know s.th. hence it is true
- Serial
  there is always a future
- Symmetric
- Equivalence (*universal*)
- Confluent *(Church-Rosser)*
- ...

Two or more:

- R_I included in R_J
- $R_I = R_J \cup R_K$
- $R_J = (R_I)^{-1}$
- $R_J = (R_I)^*$
  (transitive closure)
- $R_I \circ R_J = R_J \circ R_I$
- Confluent
- ...

Constraints on R

One relation:

- Transitive
  future of future is future
- Reflexive
  I know s.th. hence it is true
- Serial
  there is always a future
- Symmetric
- Equivalence (*universal*)
- Confluent *(Church-Rosser)*
- ...

Two or more:

- R_I included in R_J
- $R_I = R_J \cup R_K$
- $R_J = (R_I)^{-1}$
- $R_J = (R_I)^*$
  (transitive closure)
- $R_I \circ R_J = R_J \circ R_I$
- Confluent
- ...

Constraints on R

One relation:

- Transitive
  future of future is future
- Reflexive
  I know s.th. hence it is true
- Serial
  there is always a future
- Symmetric
- Equivalence (*universal*)
- Confluent *(Church-Rosser)*
- ...

Two or more:

- R_I included in R_J
- $R_I = R_J \cup R_K$
- $R_J = (R_I)^{-1}$
- $R_J = (R_I)^*$
  (transitive closure)
- $R_I \circ R_J = R_J \circ R_I$
- Confluent
- ...

Constraints on R

One relation:

- Transitive
  future of future is future
- Reflexive
  I know s.th. hence it is true
- Serial
  there is always a future
- Symmetric
- Equivalence (*universal*)
- Confluent *(Church-Rosser)*
- ...

Two or more:

- R_I included in R_J
- $R_I = R_J \cup R_K$
- $R_J = (R_I)^{-1}$
- $R_J = (R_I)^*$
  (transitive closure)
- $R_I \circ R_J = R_J \circ R_I$
- Confluent
- ...

Constraints on V

- HL(@): a nominal is true at a unique world
- Intuitionistic: atoms persist along paths
- PAL: literals persist throughout (updated) models
- ...

Interests

Given		Question
Property P	System S	does S have P?
Property P		is there a system S having P?
Property P		which system S may have P?

Problems

	Input		Output
	Formula	Model	
Model Checking	x	x	Yes/No
Satisfiability / Validity	x		Yes/No
Model Construction	x		Model/Counter-Model

Methods

Since 1950's...

- Sequent calculi [Beth, Gentzen, . . .]
  - Proof search
- Tableau calculi [Smullyan, Fitting, Massacci, . . .]
  - Model existence check
- Tableau by graph rewriting [à la toulousaine]
  - Model construction

Example

a node with the input formula

[] P & <> Q & <> (R v ~ P)

Example

$M, w \Vdash A \wedge B$ iff $M, w \Vdash A$ and $M, w \Vdash B$

A B

[] P & <> Q & <> (R v ~ P)

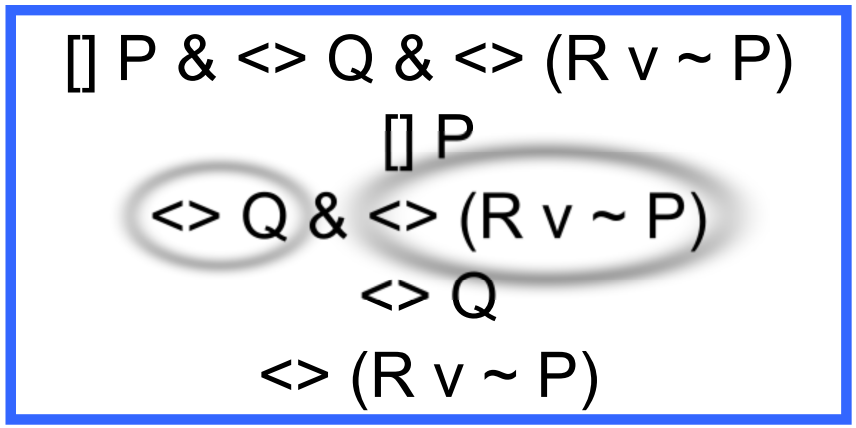
Example

$M, w \Vdash A \wedge B$ iff $M, w \Vdash A$ and $M, w \Vdash B$

[] P & <> Q & <> (R v ~ P)
[] P
<> Q & <> (R v ~ P)

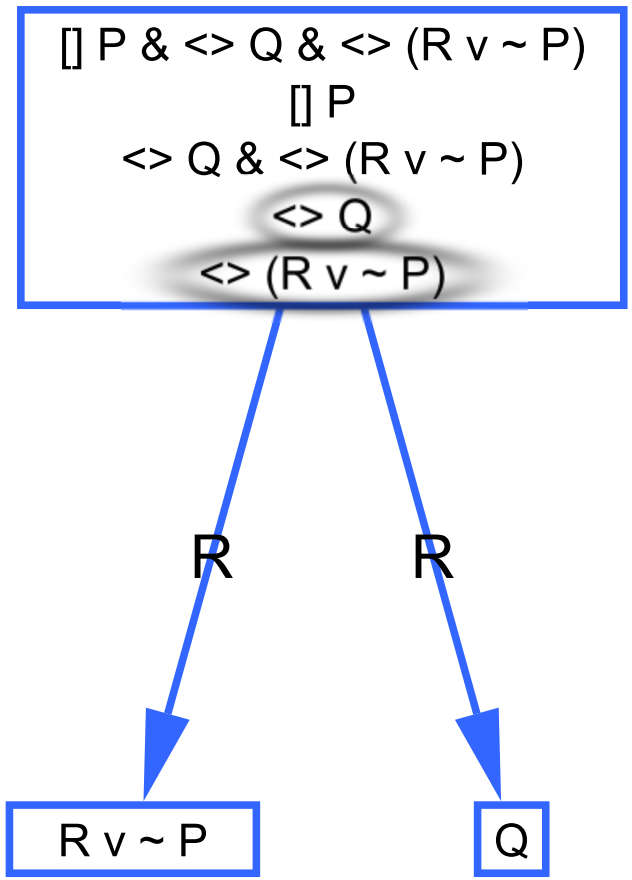
Example

$M, w \Vdash A \wedge B$ iff $M, w \Vdash A$ and $M, w \Vdash B$



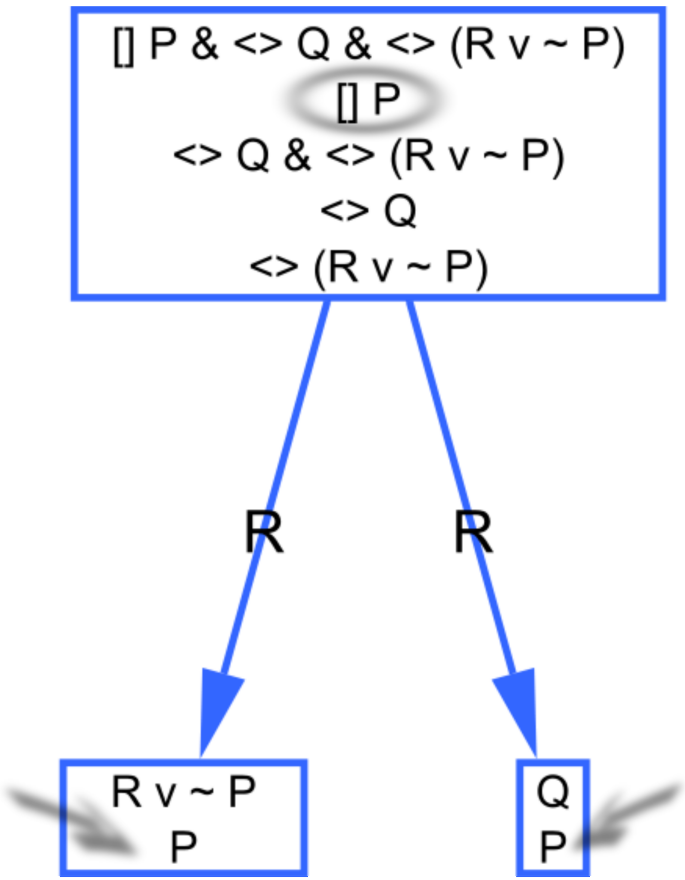
Example

$$M, w \Vdash \Diamond A \quad \text{iff} \quad \exists u \mid wRu \quad \text{and} \quad M, u \Vdash A$$



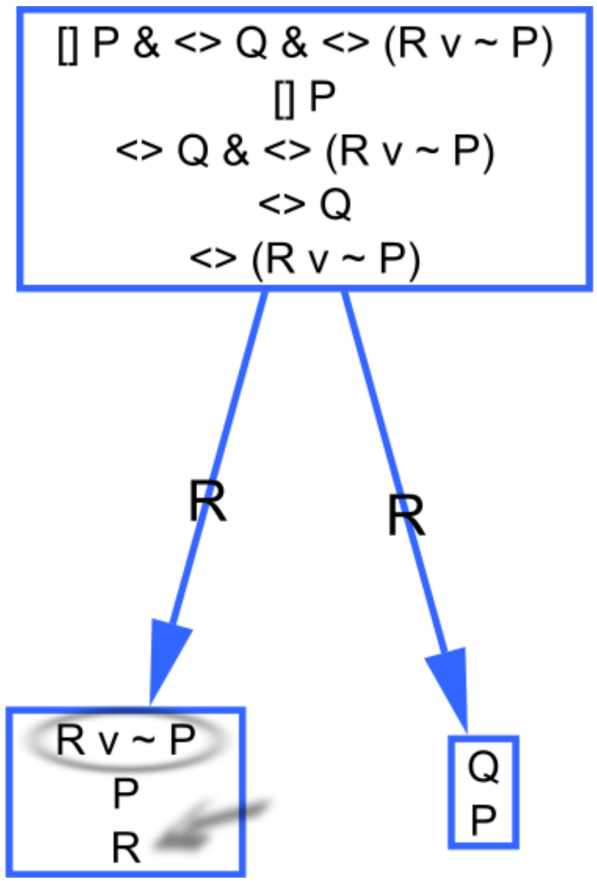
Example

$M, w \Vdash \Box A$ iff $\forall u : wRu$ then $M, u \Vdash A$

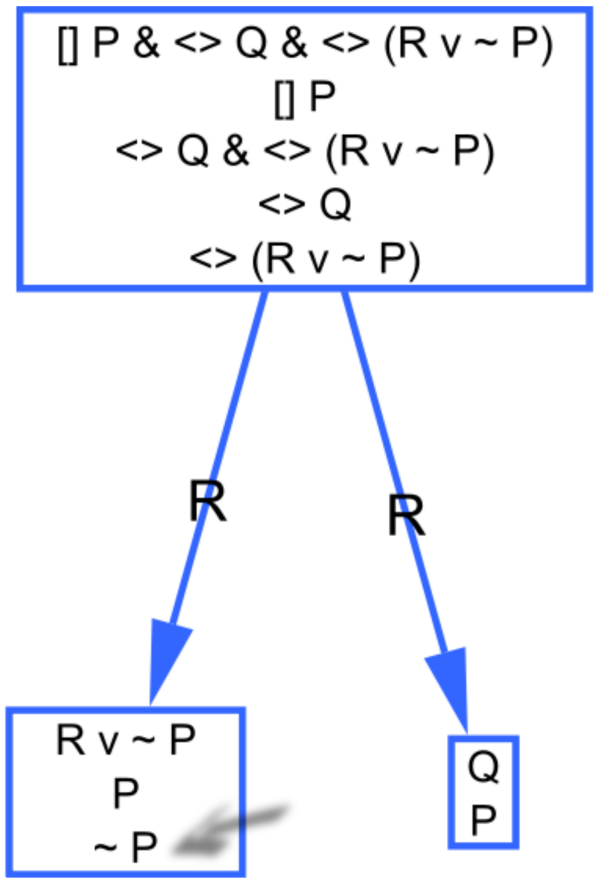


Example

$M, w \Vdash A \vee B$ iff $M, w \Vdash A$ or $M, w \Vdash B$

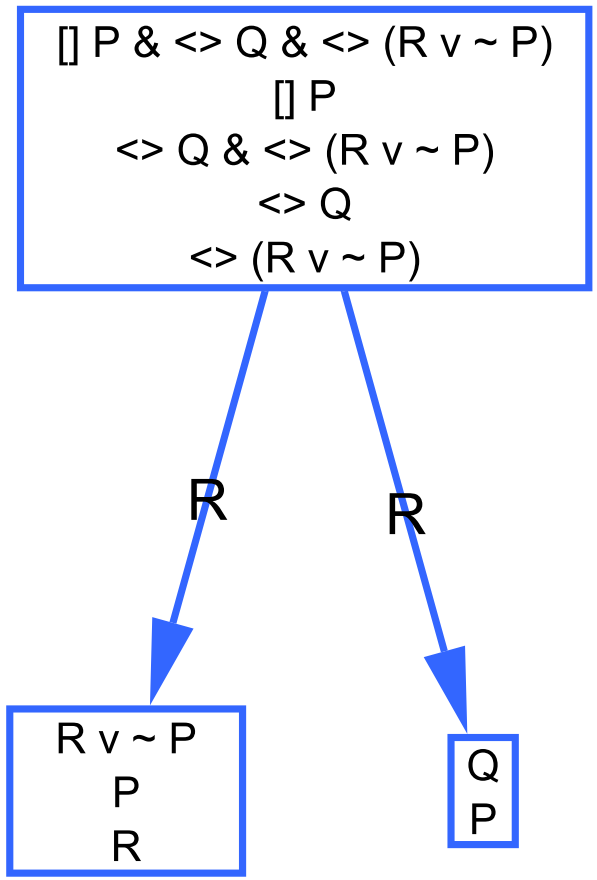


premodel 1

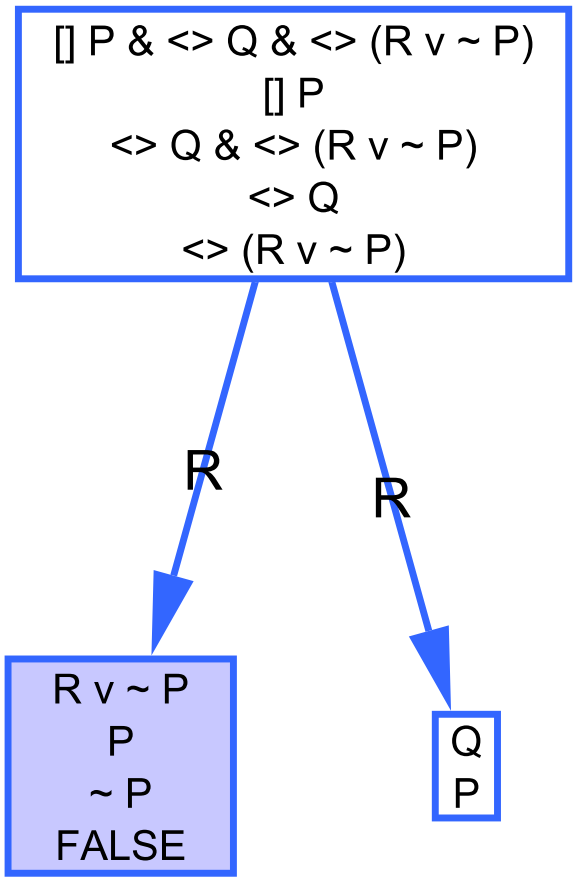


premodel 2

Example

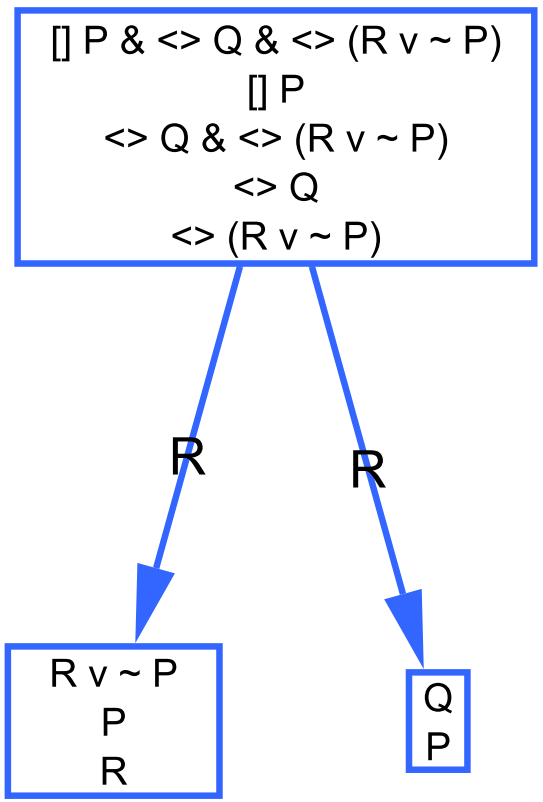


premodel 1



premodel 2

Example

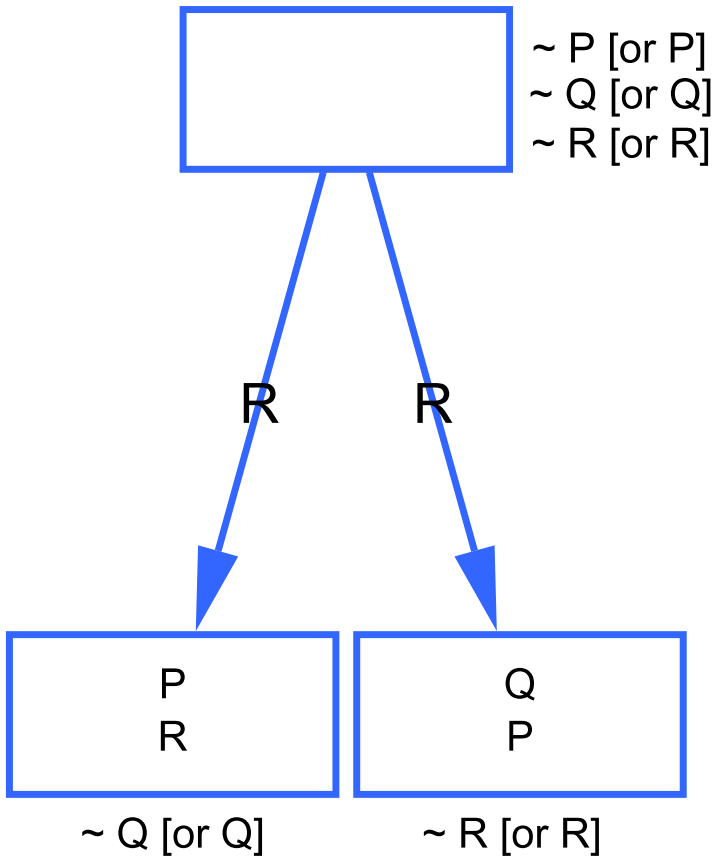


premodel 1

Model

$\Longrightarrow$

extraction



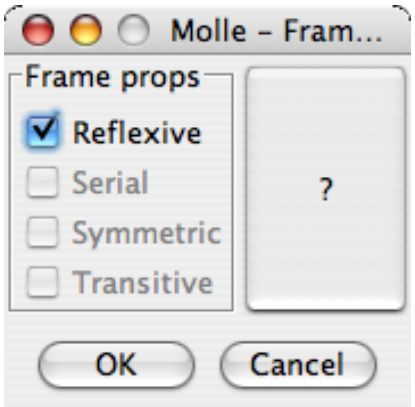
$M, w \Vdash P$ then $P \in V(w)$

Reasoning Tools

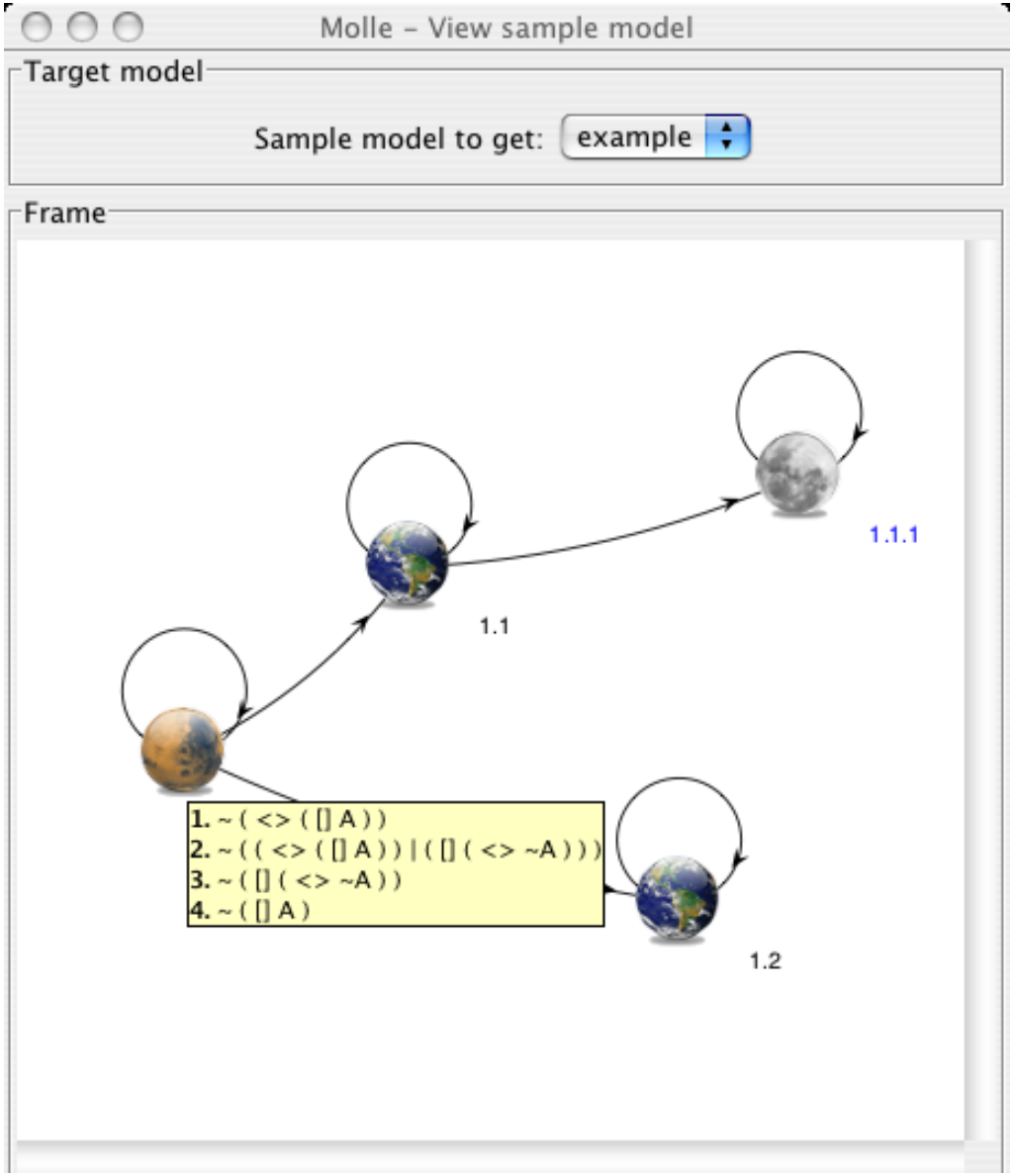
- Fast (but: geek!, not generic)
  - FaCT [Horrocks]
  - LWB [Heuerding]
  - K-SAT [Giunchiglia & Sebastiani]
- Generic (but: limited, requires coding in OCaml)
  - TWB [Abate & Goré]
- Educative (but: not generic!)
  - OOPS [Valkenhoef, Vaart & Verbrugge]
  - Molle [Mazzucchi & Mocci]

Reasoning Tools

Molle



Reflexivity only!

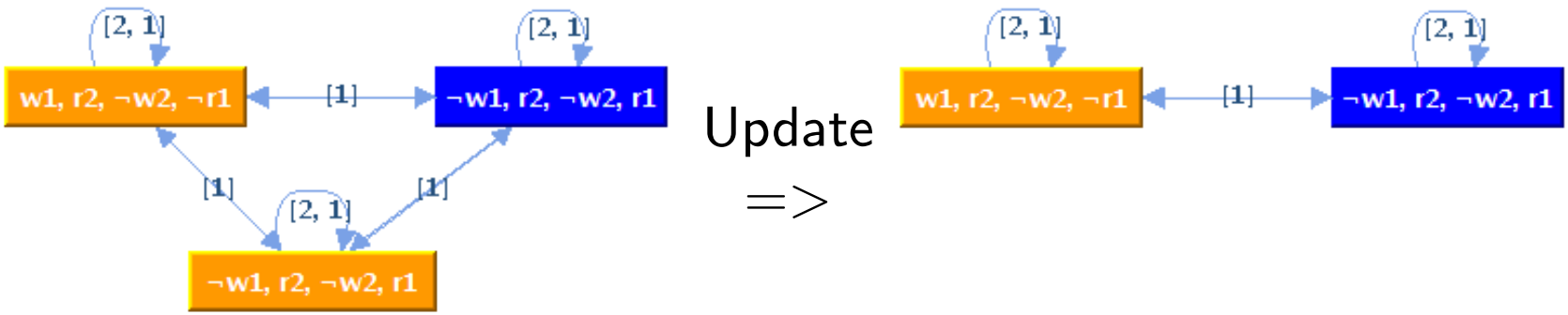


Reasoning Tools

OOPS

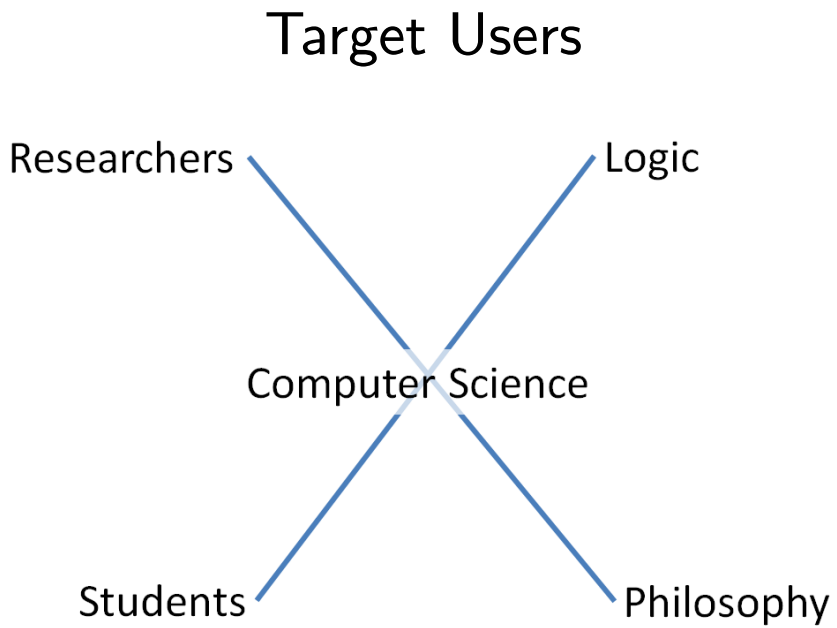
- Model update
- High-level language
- Graph viewer

- $S5_n$ only!



Desiderata

- Generic: users' own new methods
- Educative: user-friendly
- Performance: reasonable time



Outline

Uniform methodology

Semantics

via

Graph rewriting rules

How?
What are graph rewriting rules?

Uniform methodology

Semantics

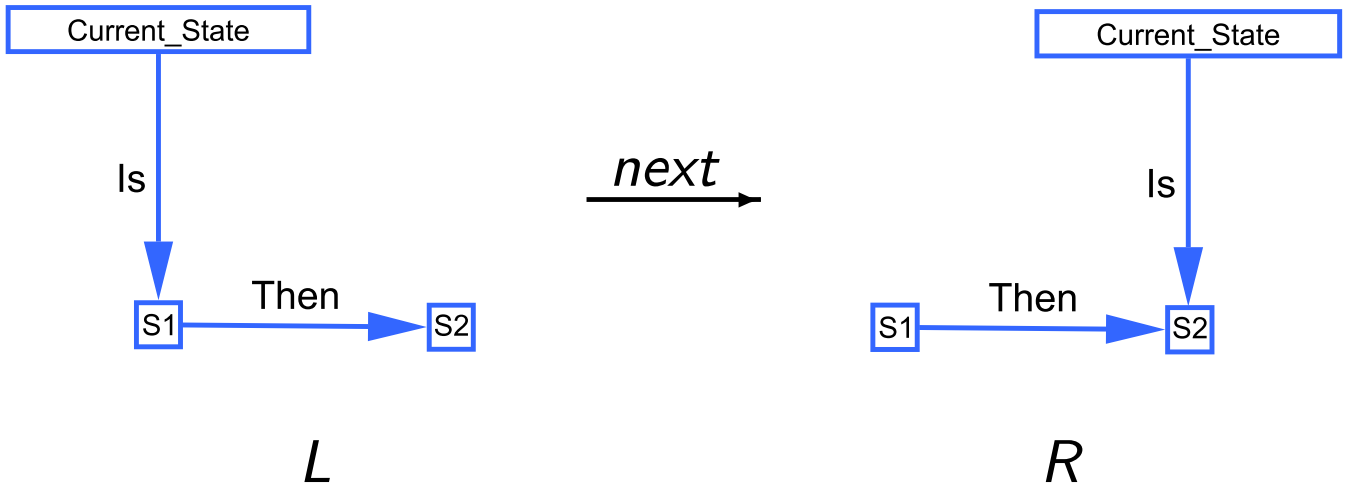
via

Graph rewriting rules

How?

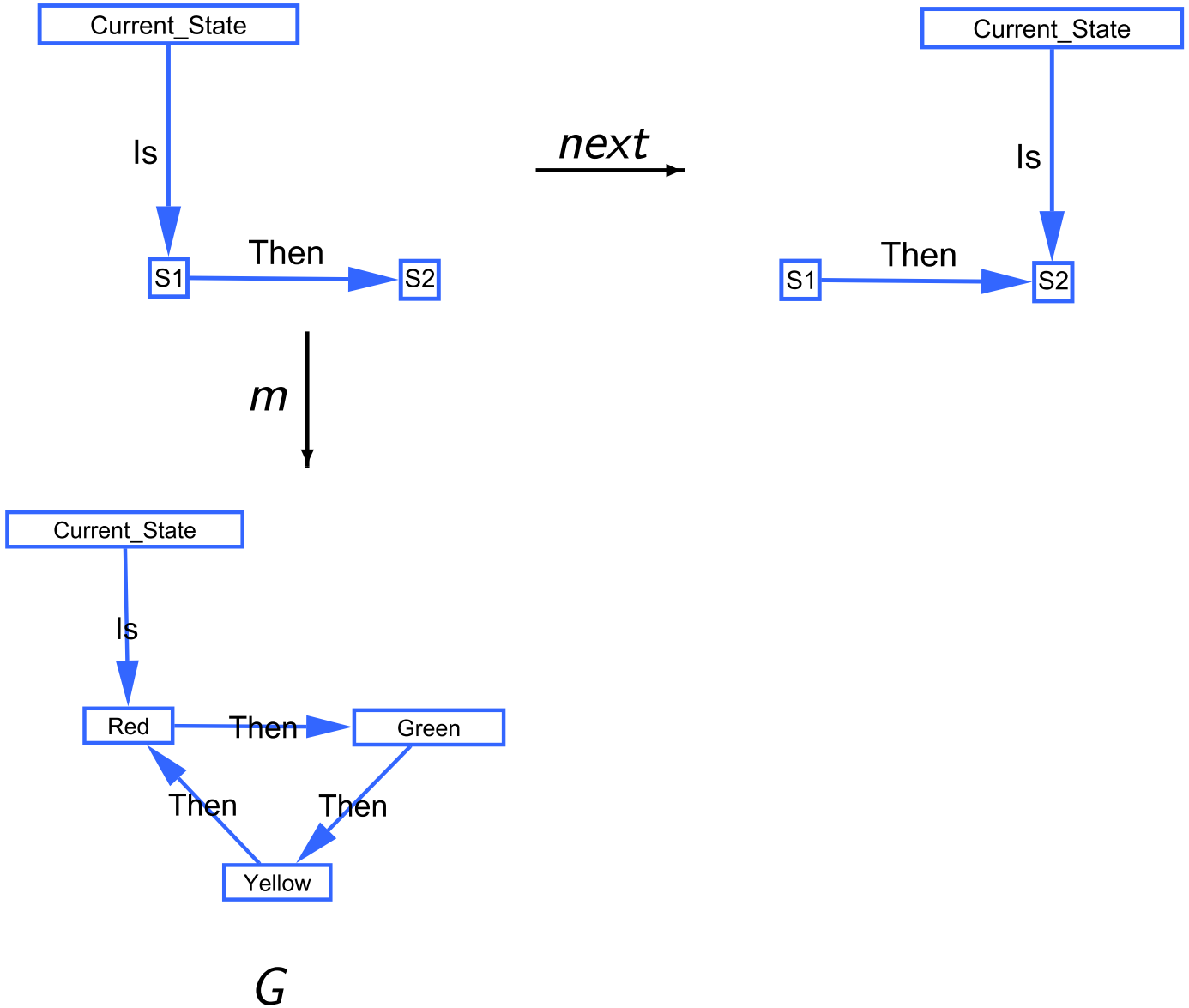
What are graph rewriting rules?

Definition

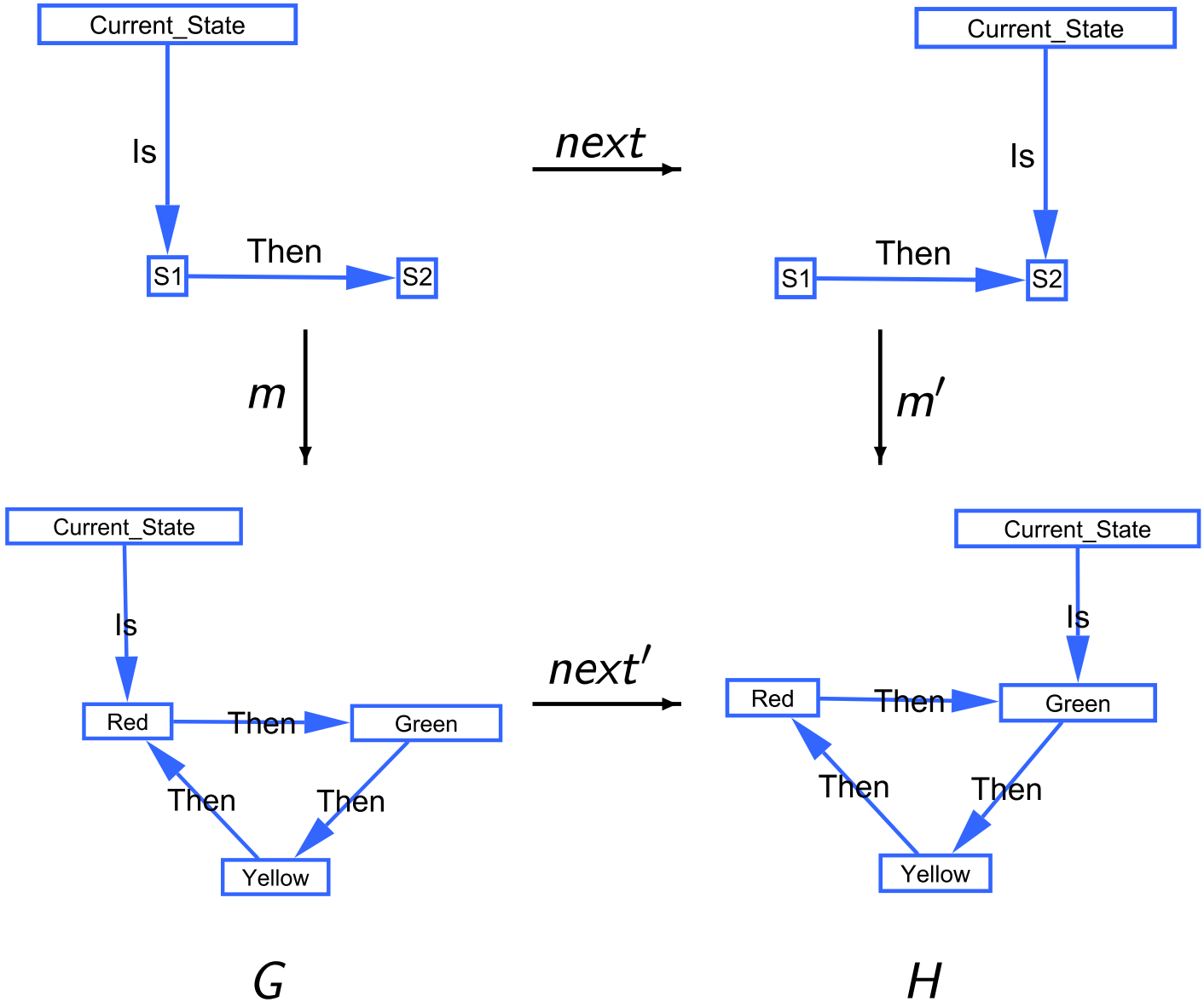


L *R*

Matching


Current_State
Is
Then
S1
S2
next
Current_State
Is
Then
S1
S2
m
Current_State
Is
Red
Then
Green
Then
Then
Yellow
G

Application



Uniform methodology

Semantics

via

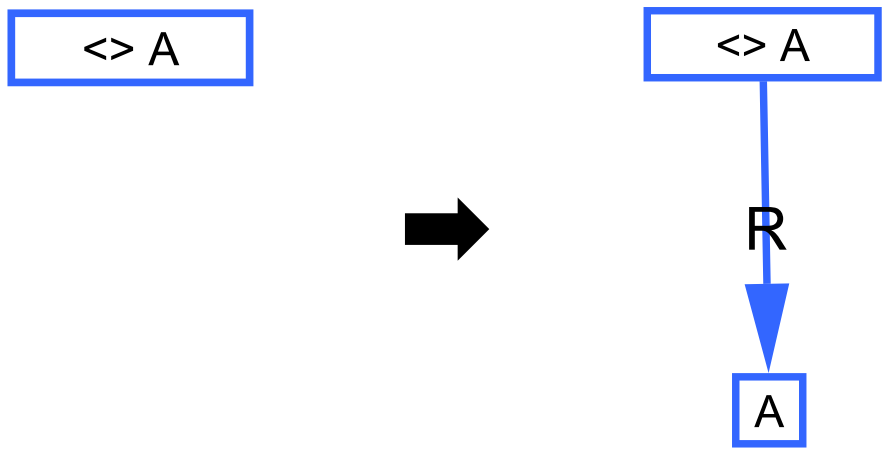
Graph rewriting rules

How?

~~What are graph rewriting rules?~~

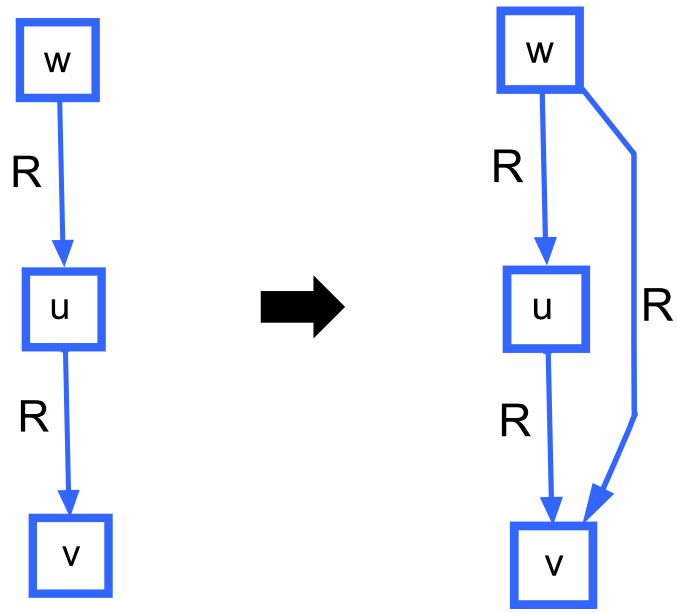
Truth conditions

$$M, w \Vdash \Diamond A \text{ iff } \exists u \mid wRu \text{ and } M, u \Vdash A$$



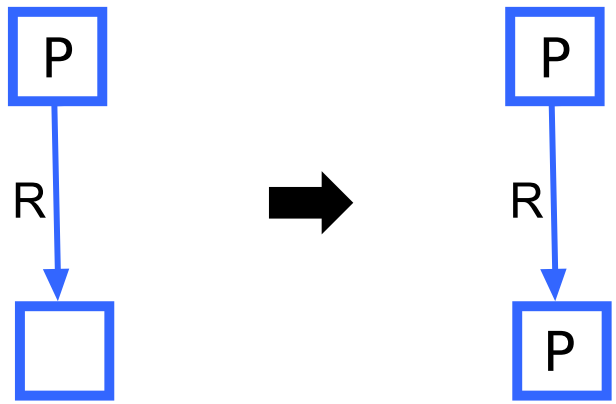
R-Contraints

Transitivity



V-Contraints

Persistence



Certifying the method

- Termination: does it halt?
- Soundness: does it consider ALL the semantics?
- Completeness: does it consider s.th. ELSE?
- Complexity: what is its time / space costs?
- ...

How to experiment with it?

User-defined language

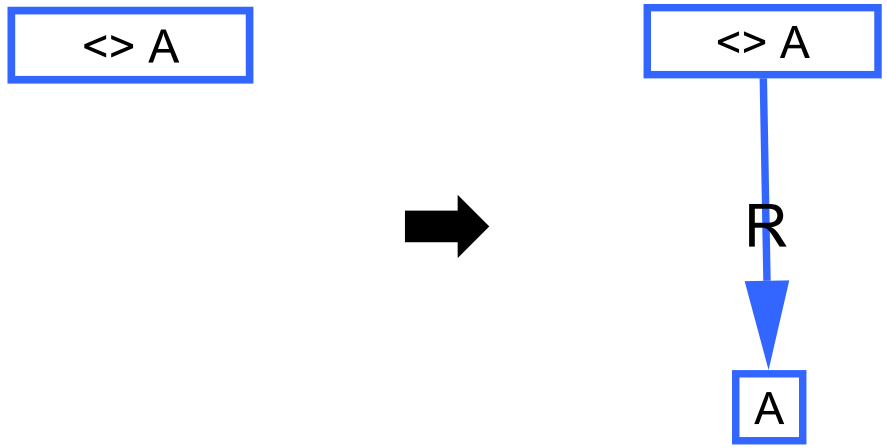
Example (definition)

name	arity	display
not	1	$\sim$ -
and	2	- & -
...		
pos	1	<> -
...		

Example (usage)

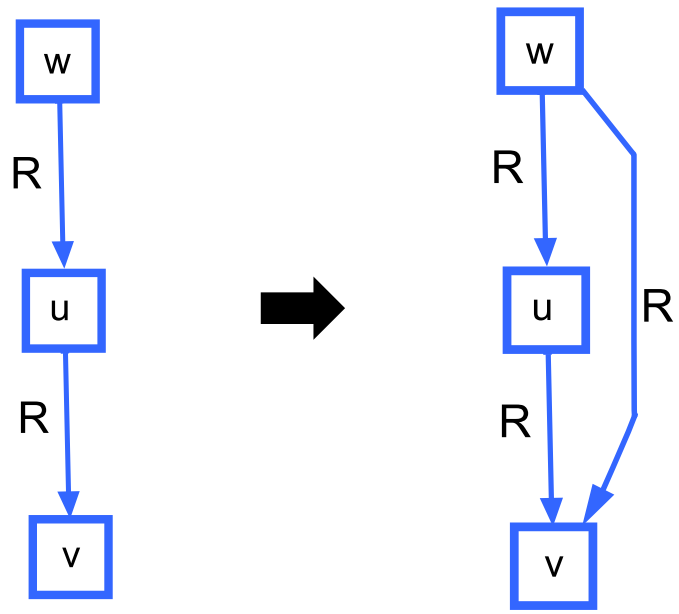
- pos P
  <> P
- and not Q not P
  $\sim$ Q & $\sim$ P

User-defined rules




```
Rule Pos
  hasElement w pos variable A

                                createNewNode u
                                link w u R
                                add u variable A

End
```

User-defined rules

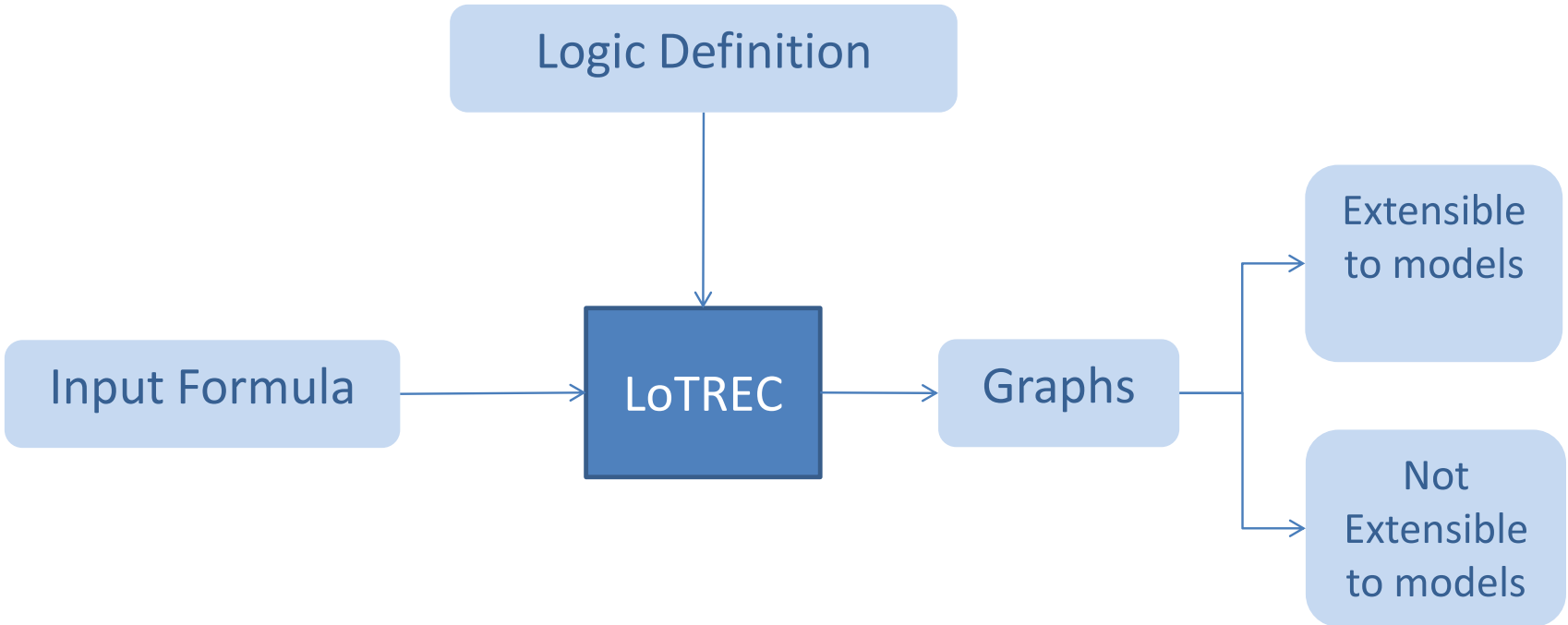



```
Rule Pos
  isLinked w u R
  isLinked u v R

                    link w u R

End
```

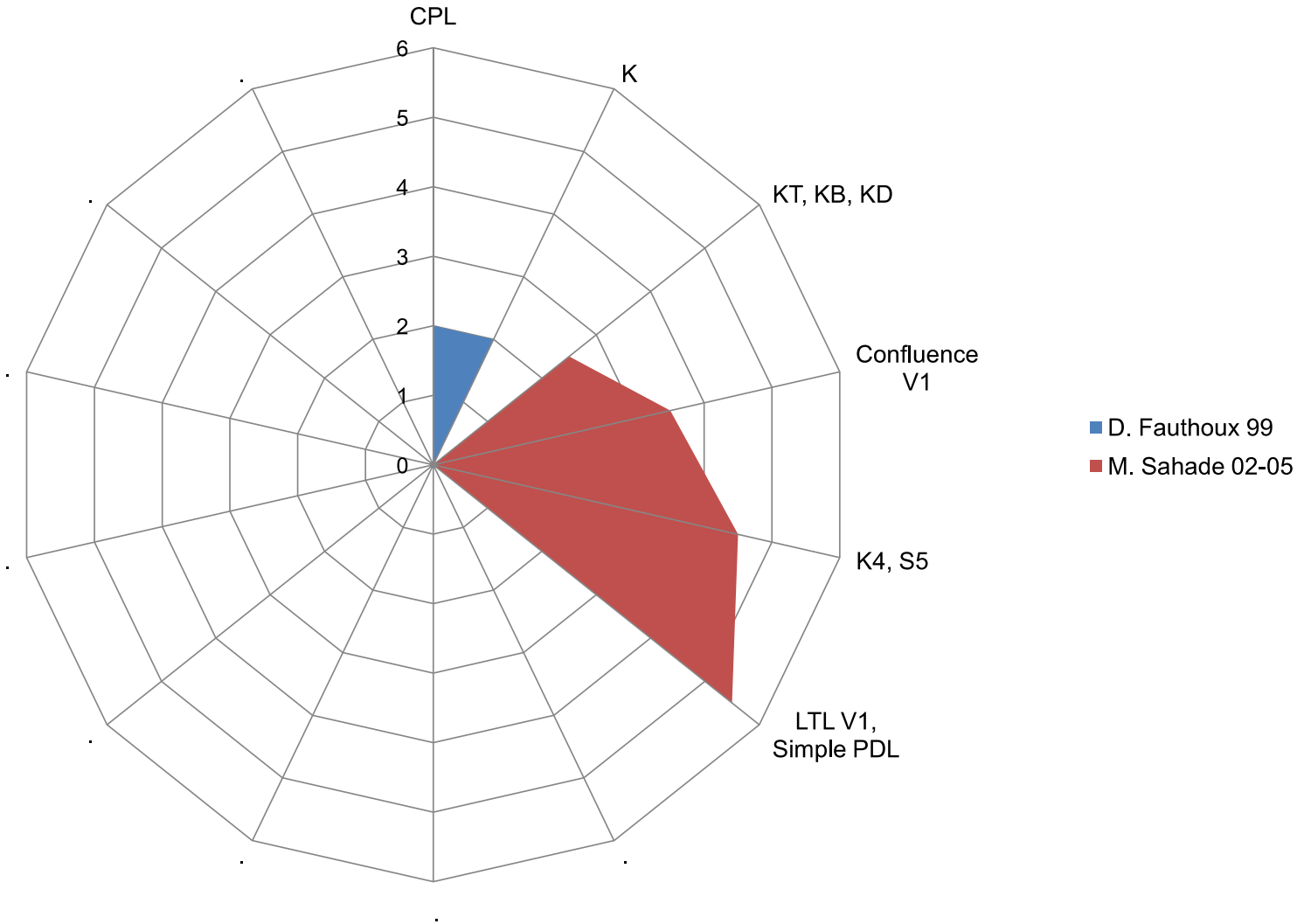
User-defined strategies

- Ordering on the rules
- Saturation (`repeat`...`end`)
- Priority (`firstRule`...`end`)

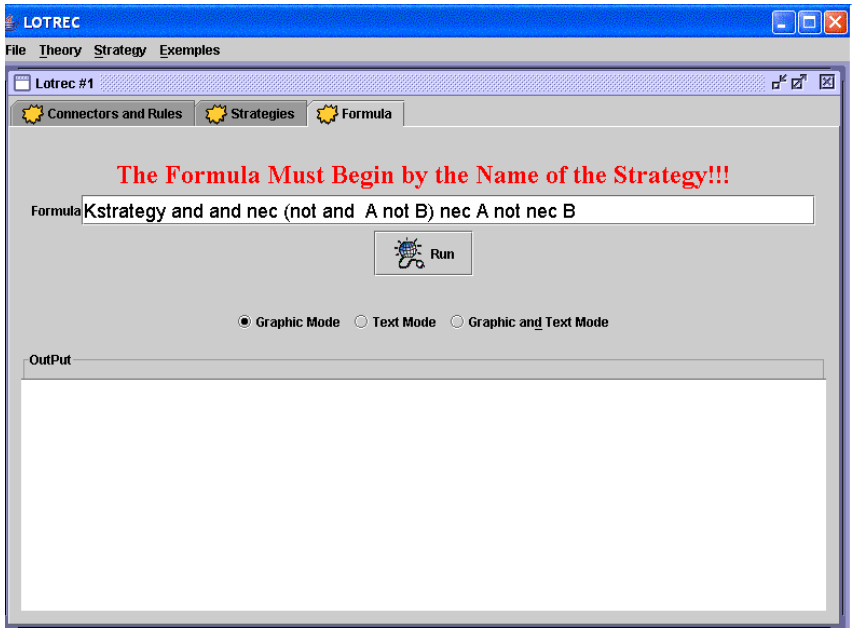
The black box

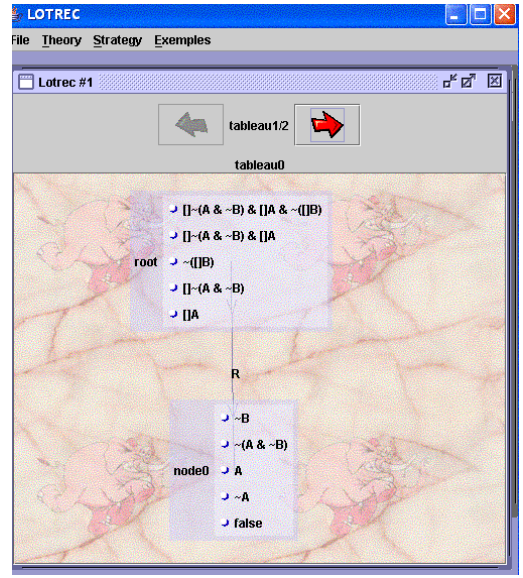


What was done?

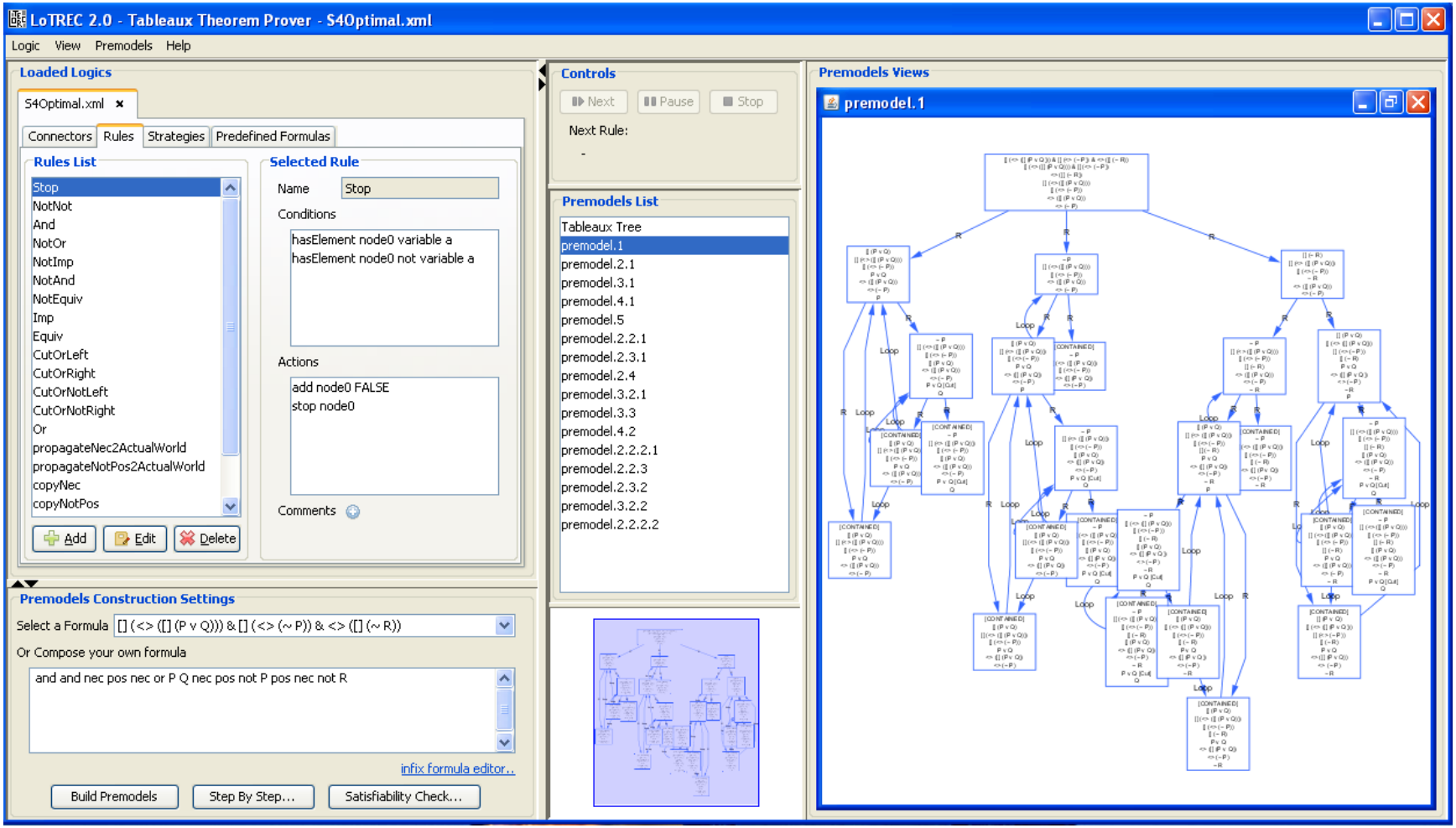


What was done?





Some of what I did

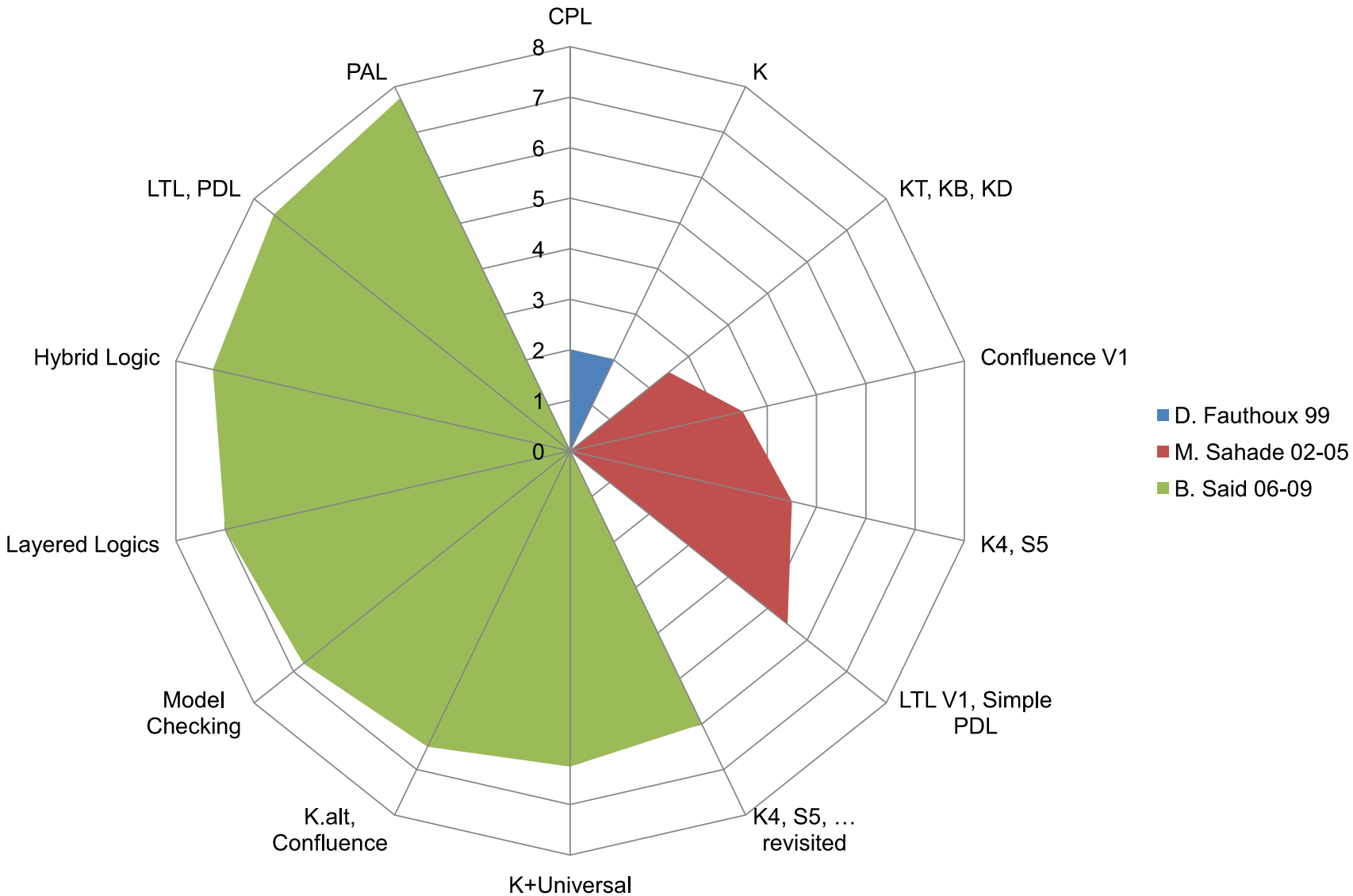


Some of what I did

Added techniques:

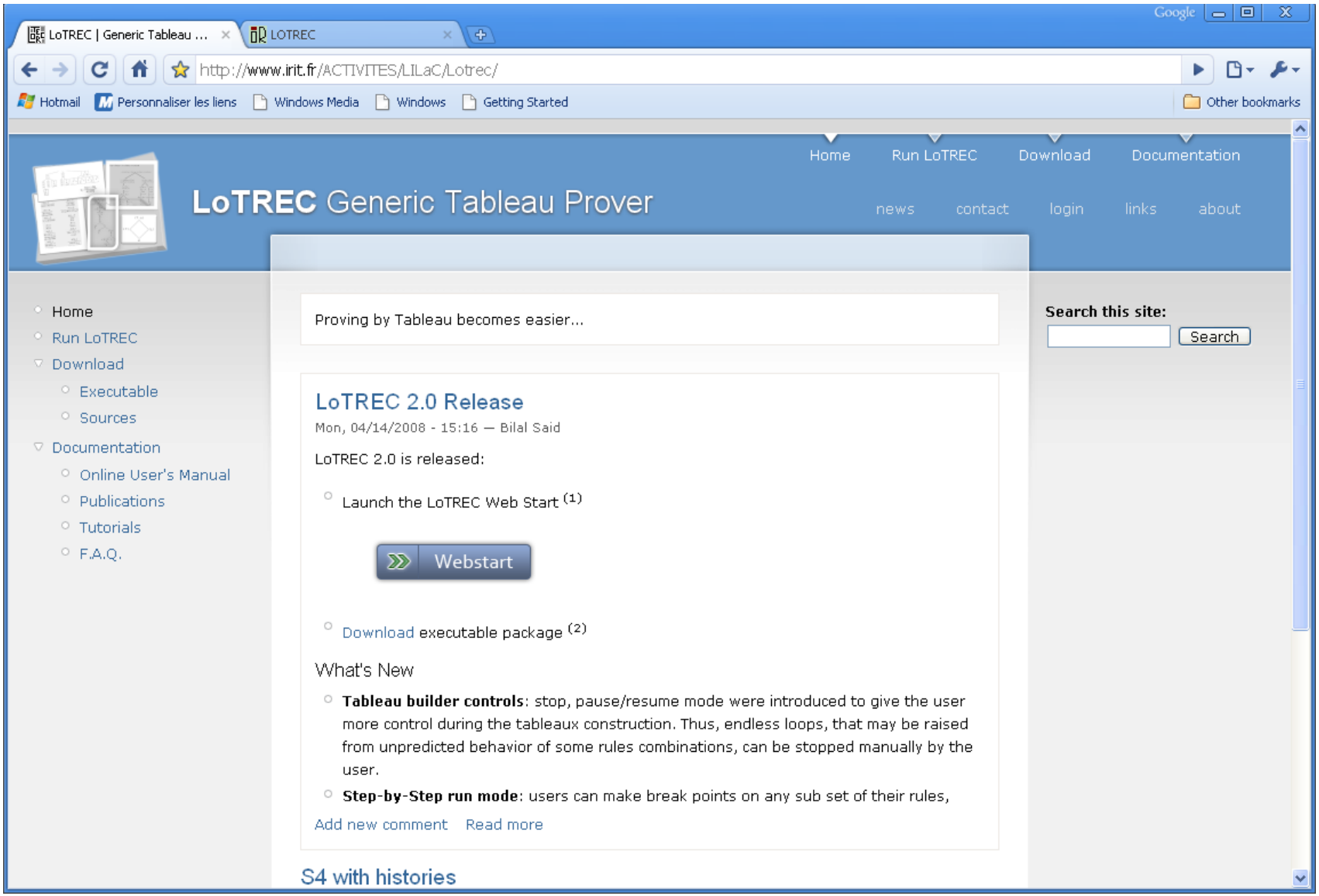
- One occurrence rules application (K.alt, LTL,...)
- Defining non-injective patterns (Confluence)
- Extend the language (Model Checking,...)
- Model checking (LTL, PDL,...)
- Labeled formulas (PAL)
- Nodes as memory cells (S4 + histories)
- Run in step-by-step mode (debugging)
- Code new extensions
- ...

Some of what I did



Running live

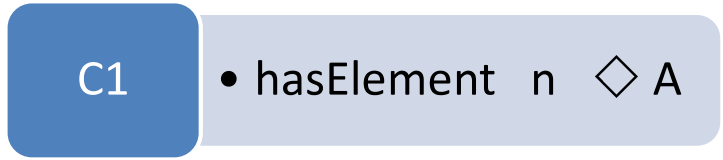
http://www.irit.fr/Lotrec

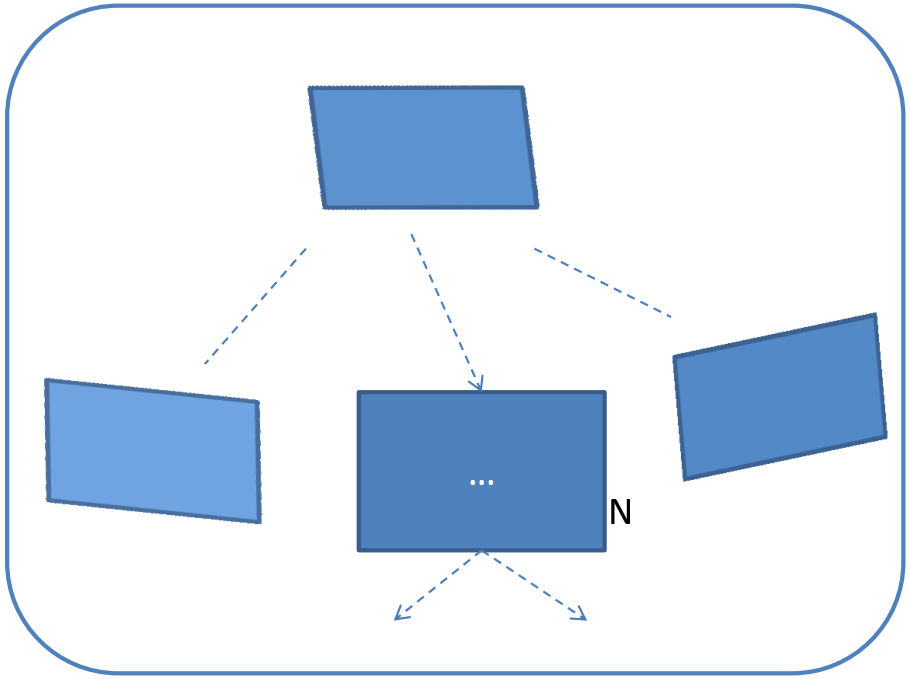


Outline

1. Modal logics
2. Model Construction
3. Event-driven pattern matching
   - Mechanism
   - Semantics
   - Evaluation

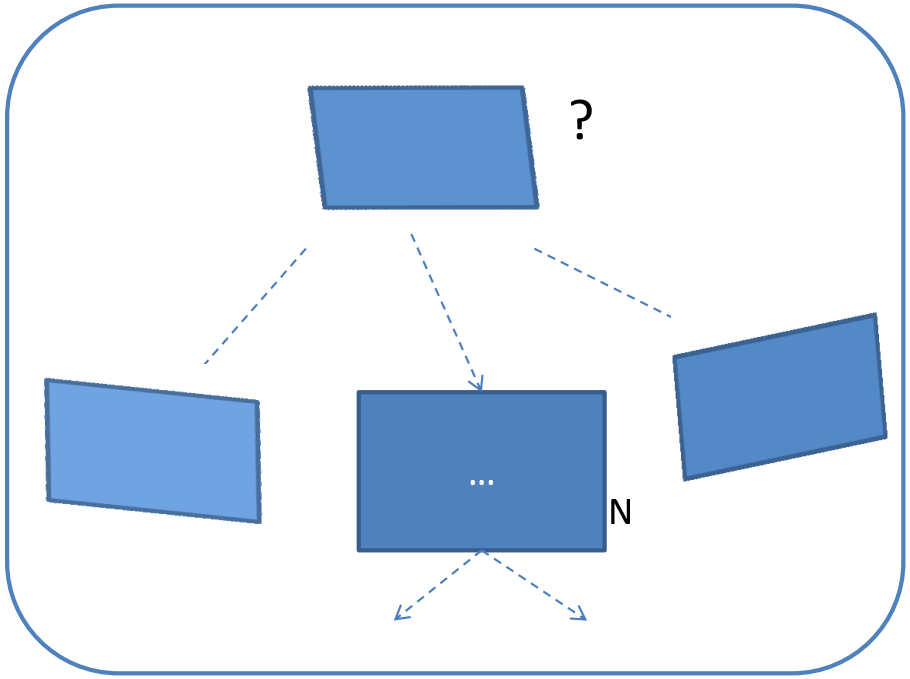
Naive pattern matching



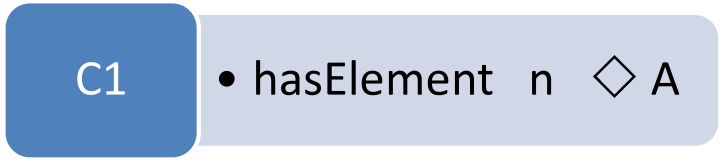


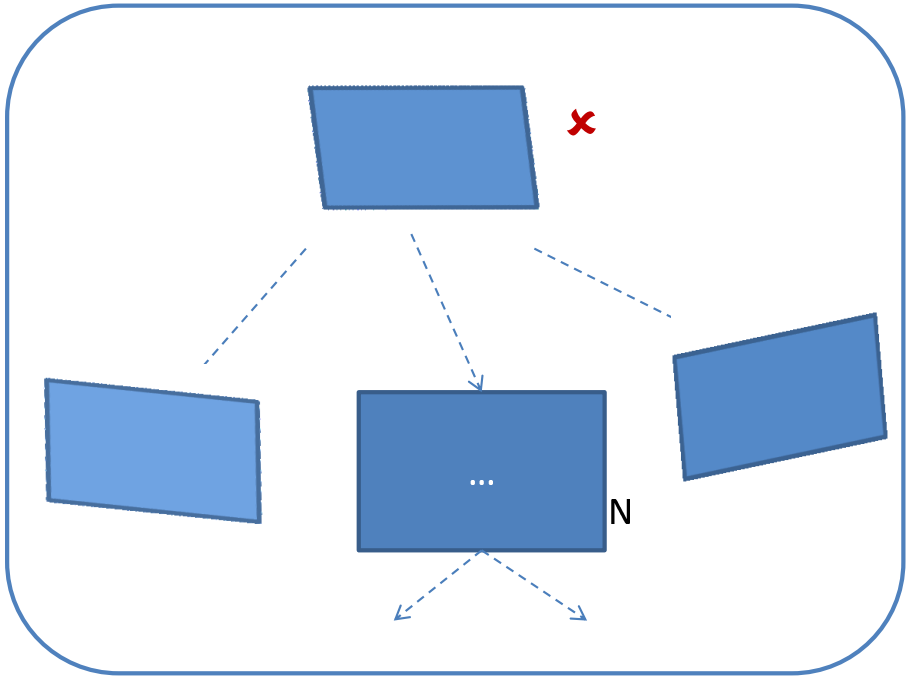
Naive pattern matching

C1 • hasElement n ◇ A



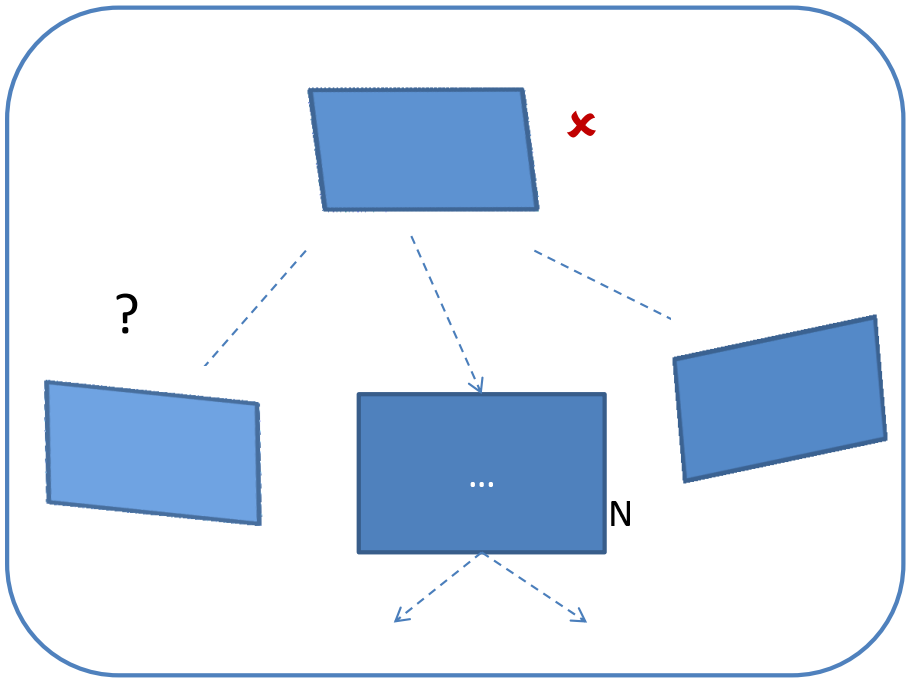
Naive pattern matching



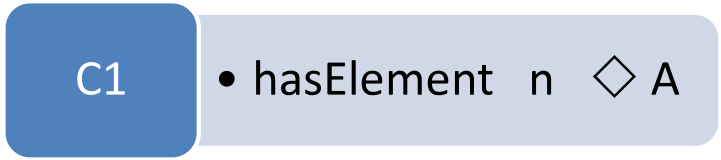


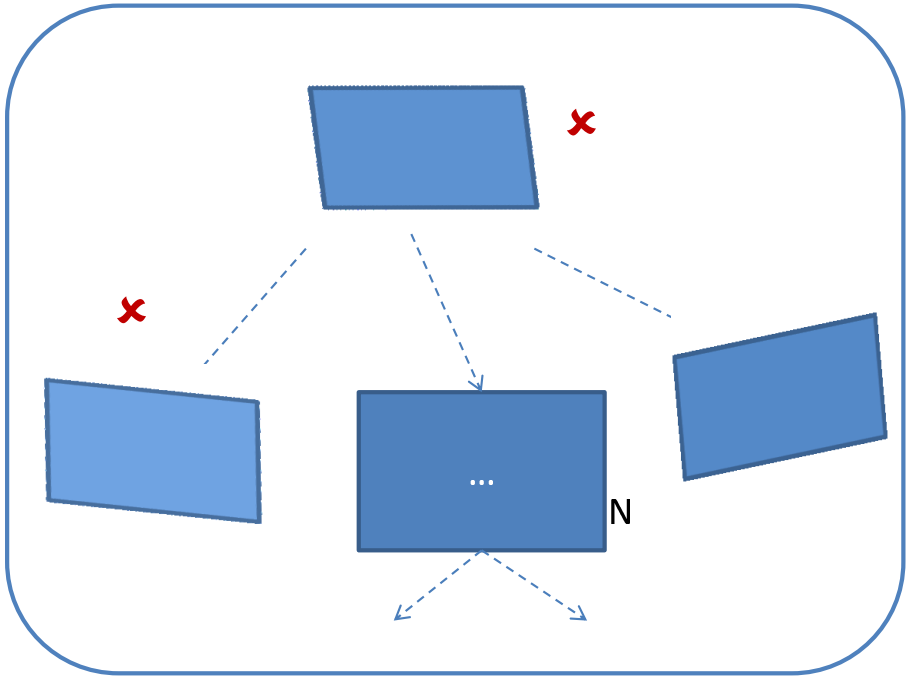
Naive pattern matching

C1 • hasElement n ◇ A



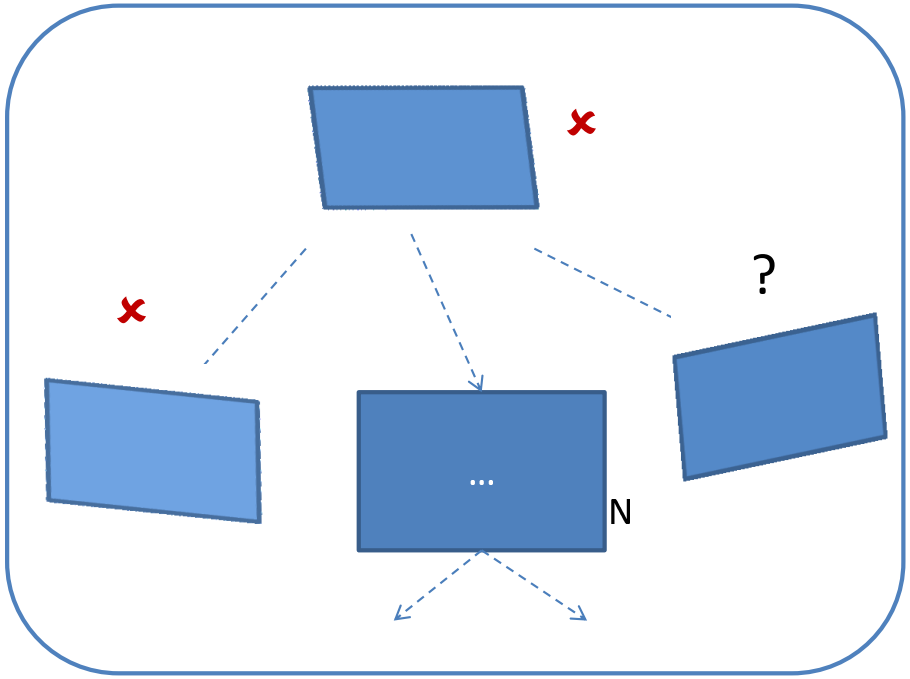
Naive pattern matching



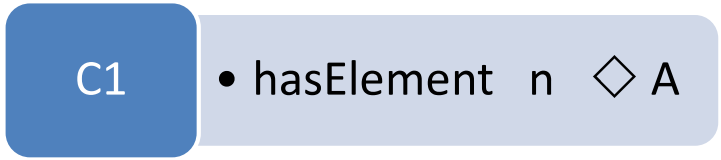


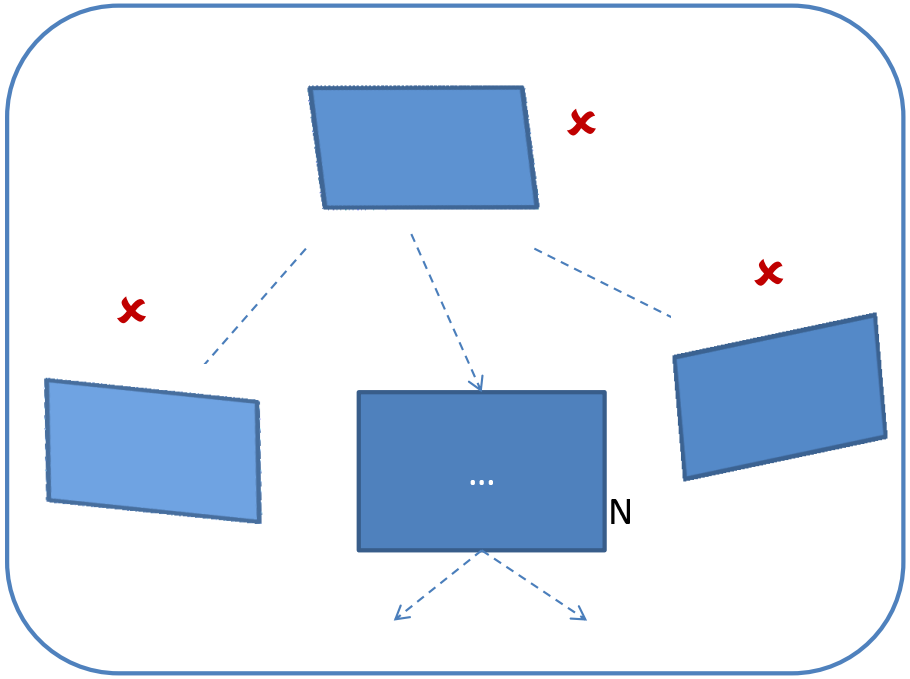
Naive pattern matching

C1 • hasElement n ◇ A



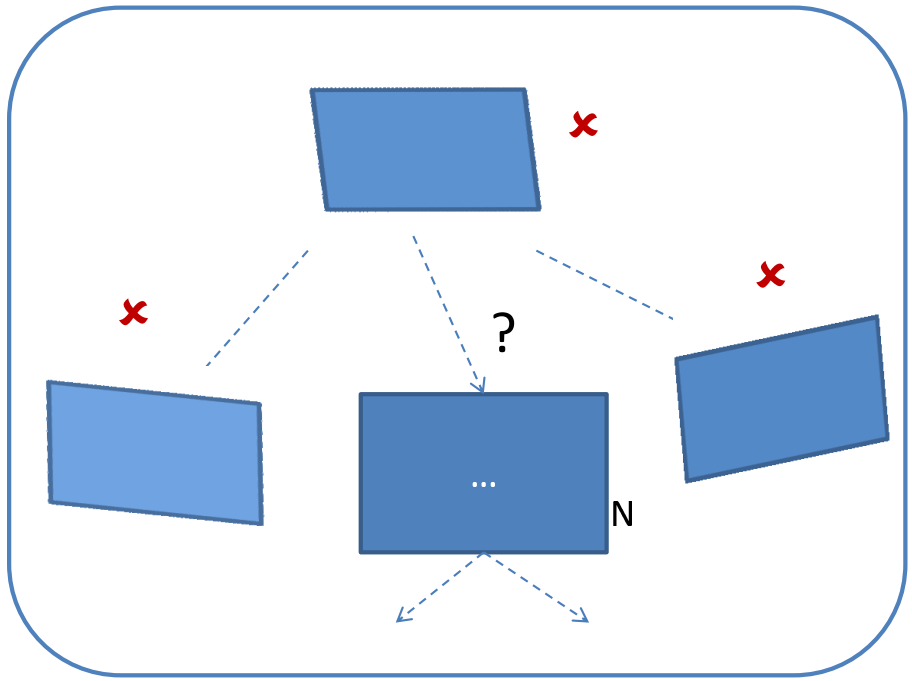
Naive pattern matching





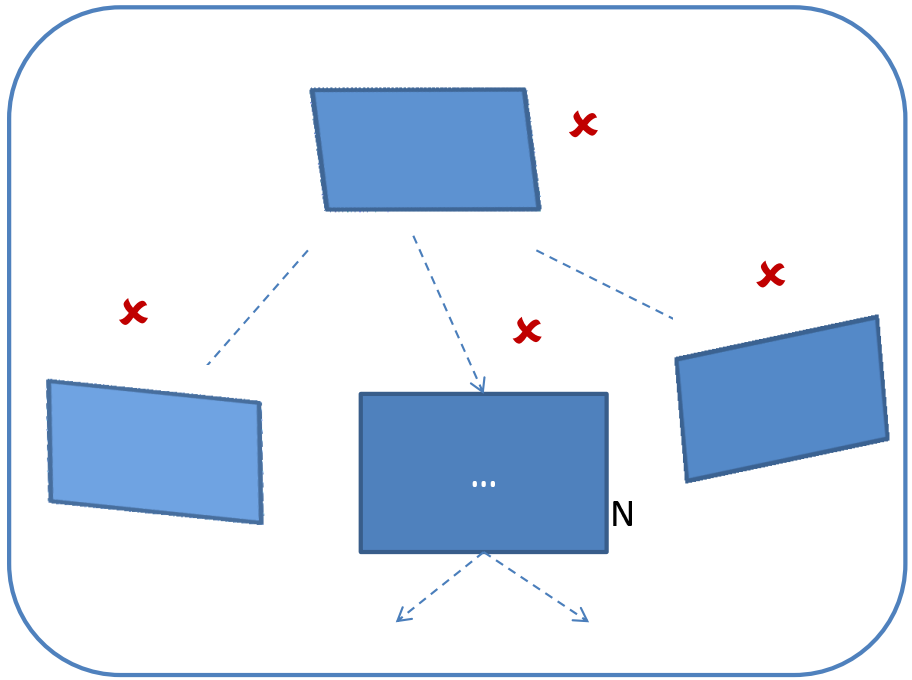
Naive pattern matching

C1 • hasElement n ◇ A



Naive pattern matching

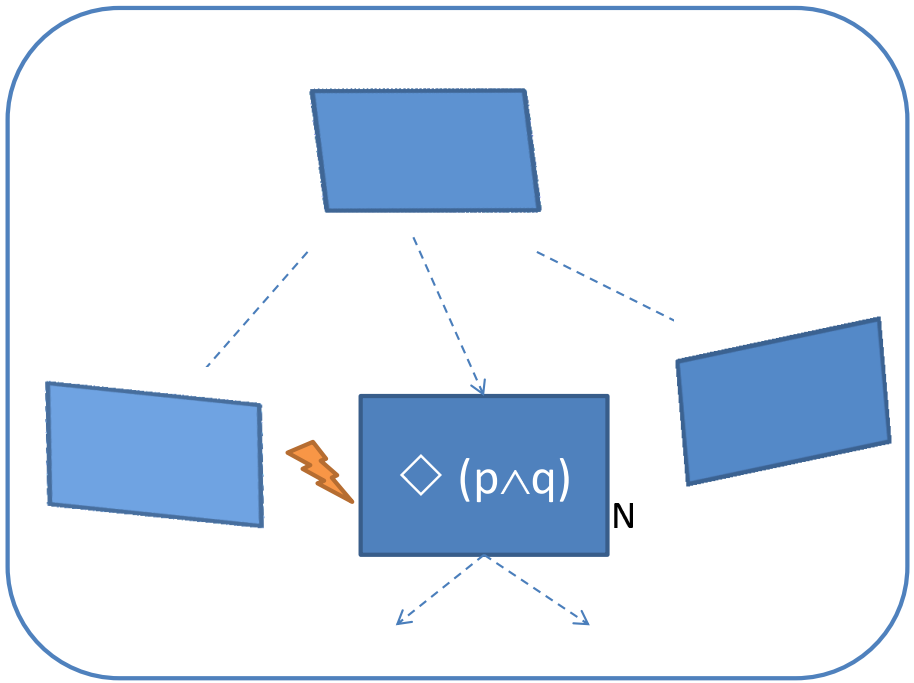
C1 • hasElement n ◇ A



$$O(|G|^{|L|})$$

Naive pattern matching

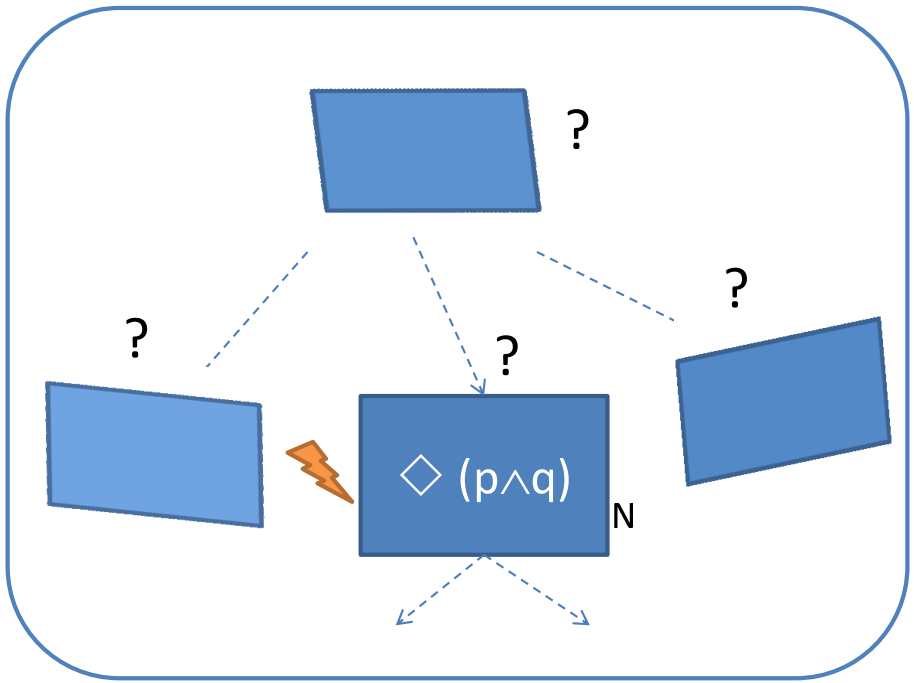
C1 • hasElement n ◇ A



$$O(|G|^{|L|})$$

Naive pattern matching

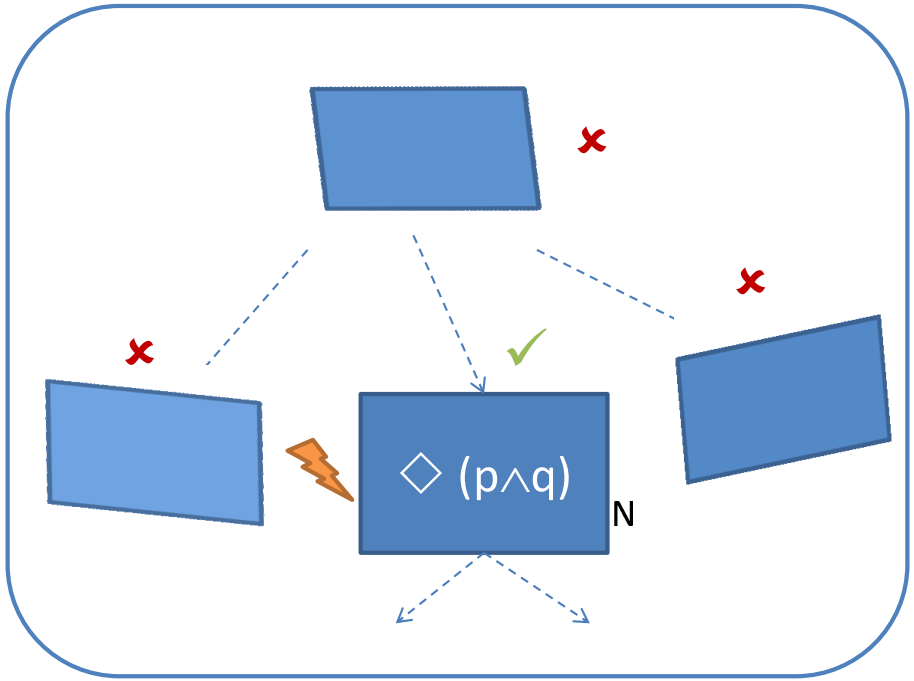
C1 • hasElement n ◇ A



$$O(|G|^{|L|})$$

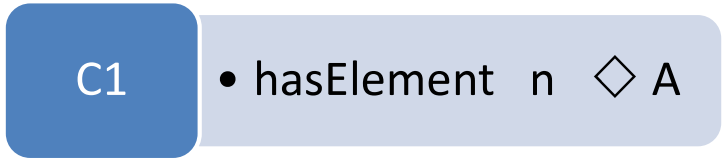
Naive pattern matching

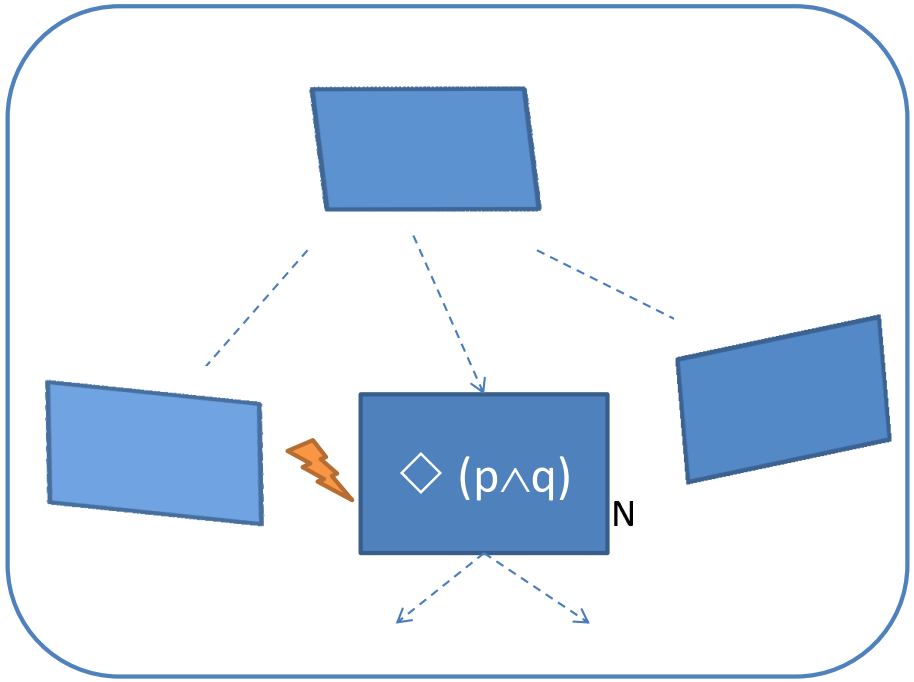
C1 • hasElement n ◇ A



$$O(|G|^{|L|})$$

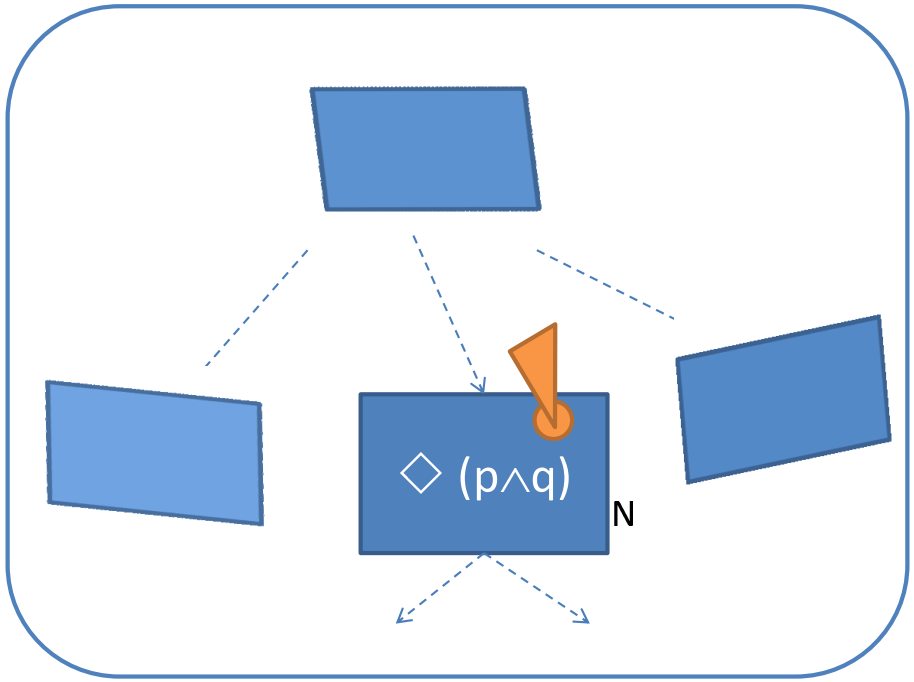
Event-based matching in LoTREC



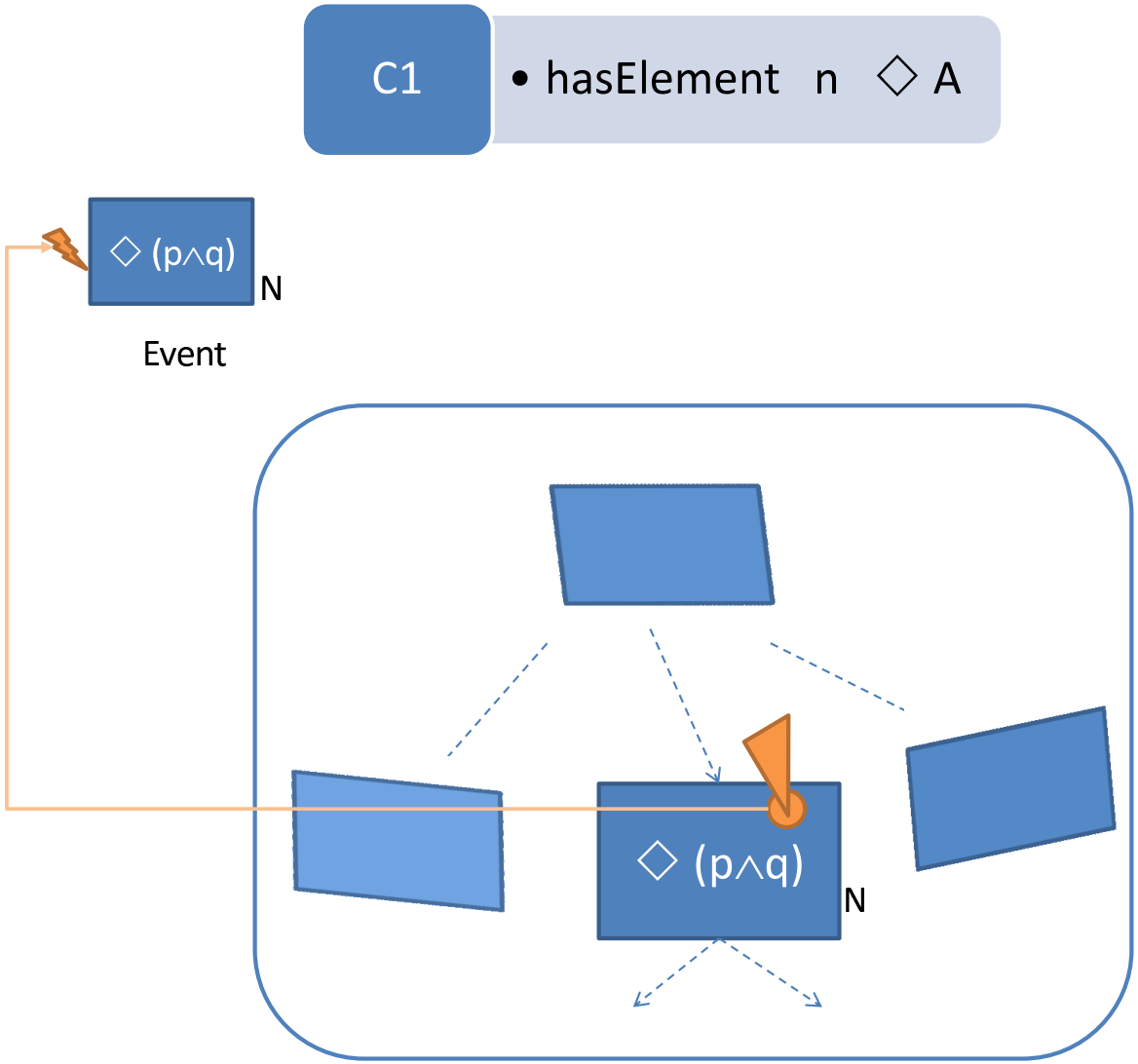


Event-based matching in LoTREC

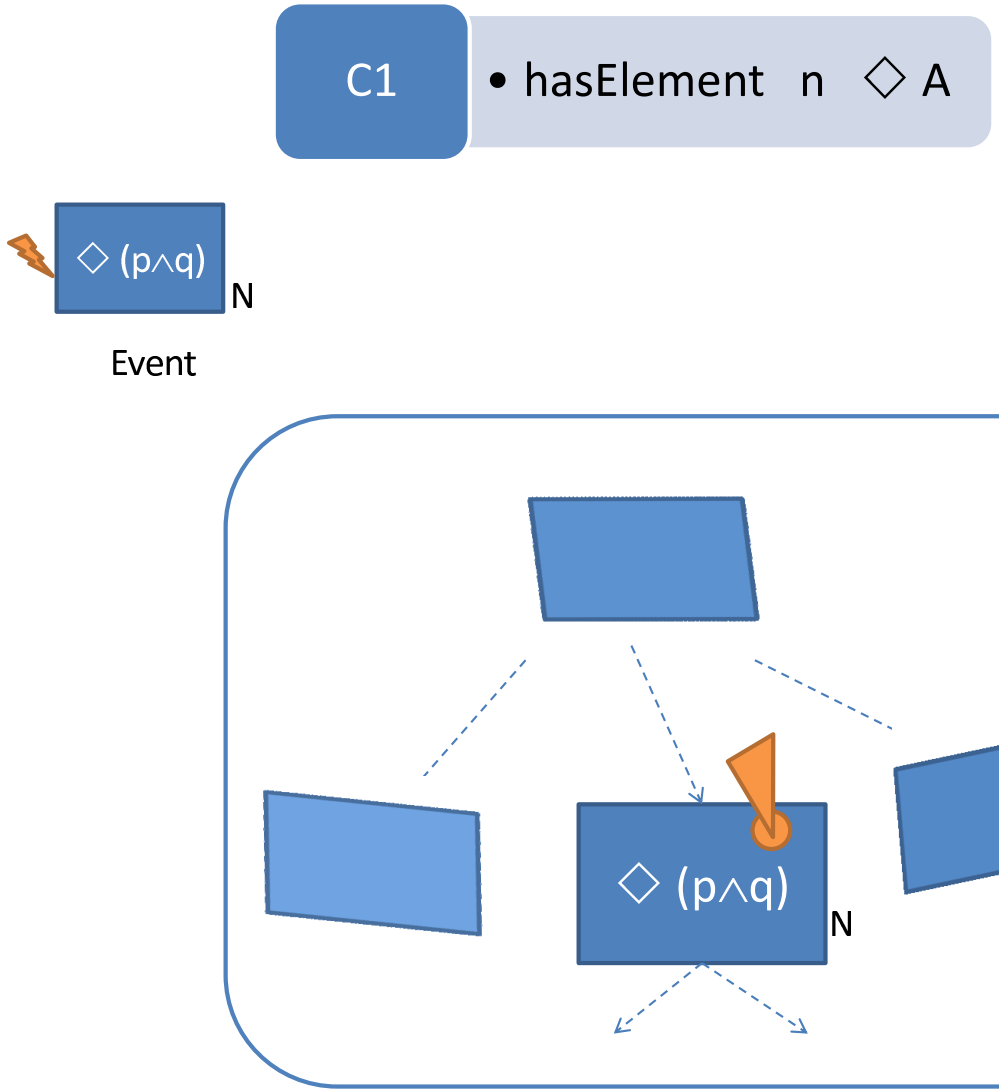
C1 • hasElement n ◇ A



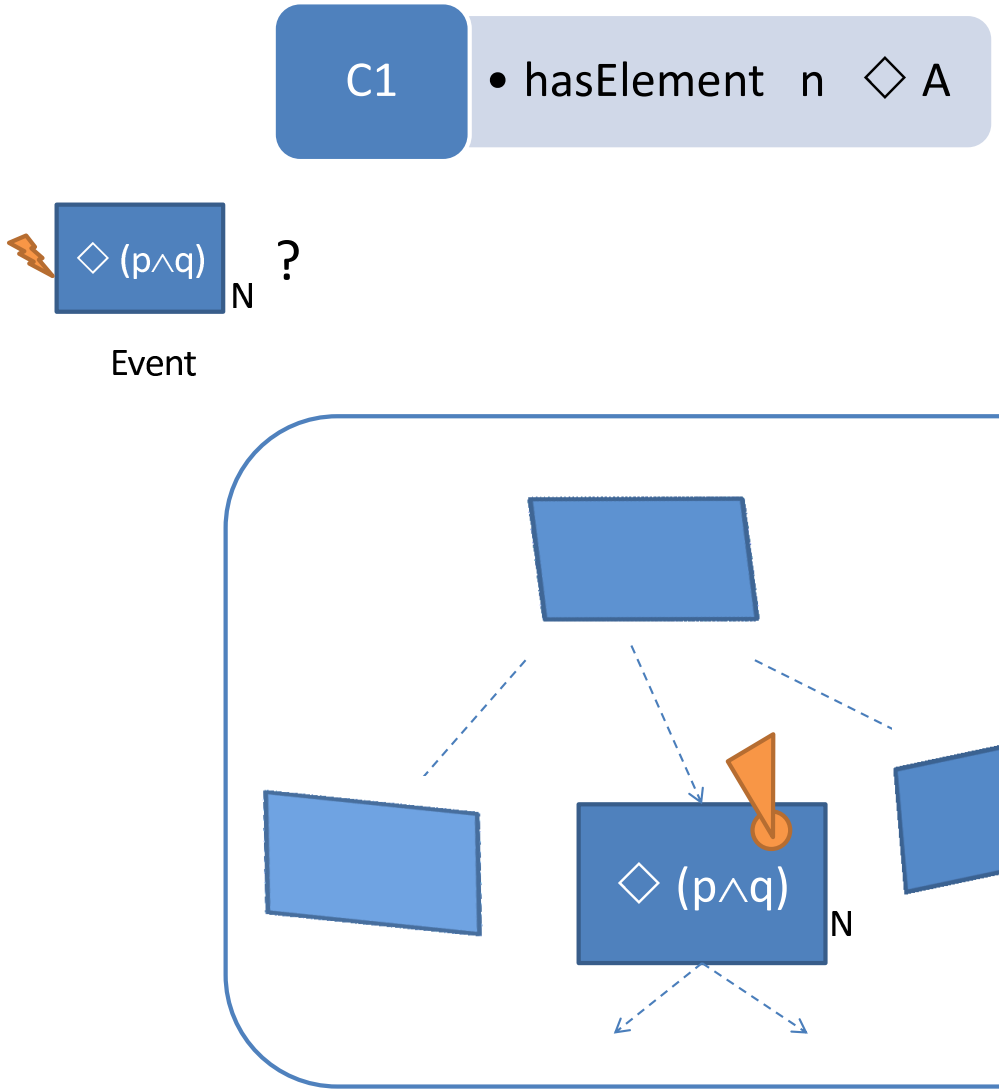
Event-based matching in LoTREC



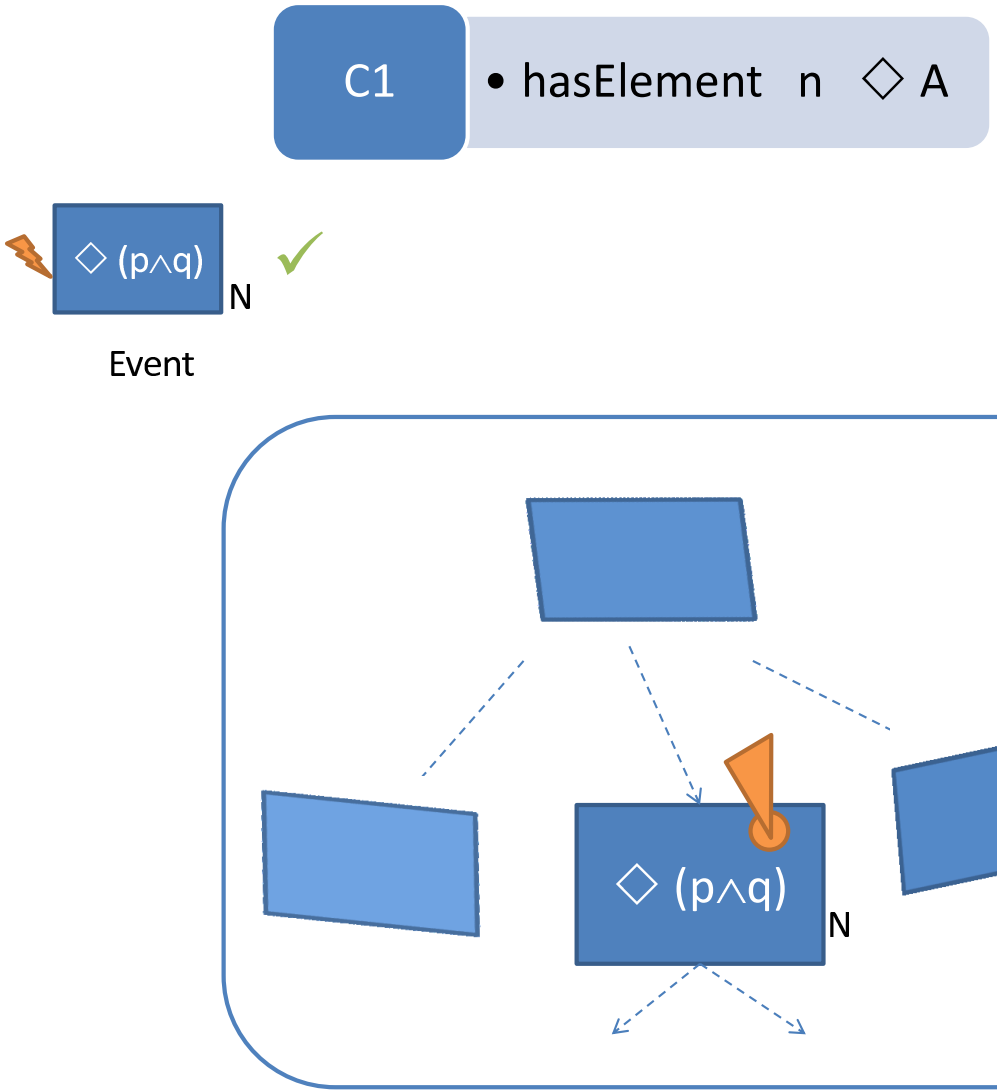
Event-based matching in LoTREC



Event-based matching in LoTREC



Event-based matching in LoTREC



$$O(k^{|L|}) \text{ where } k \ll |G|$$

Equivalence to usual semantics

[Gasquet, Said & Schwarzentruber 09]

Rewriting with event-based matching
=
Rewriting without event-based matching

Reasons:

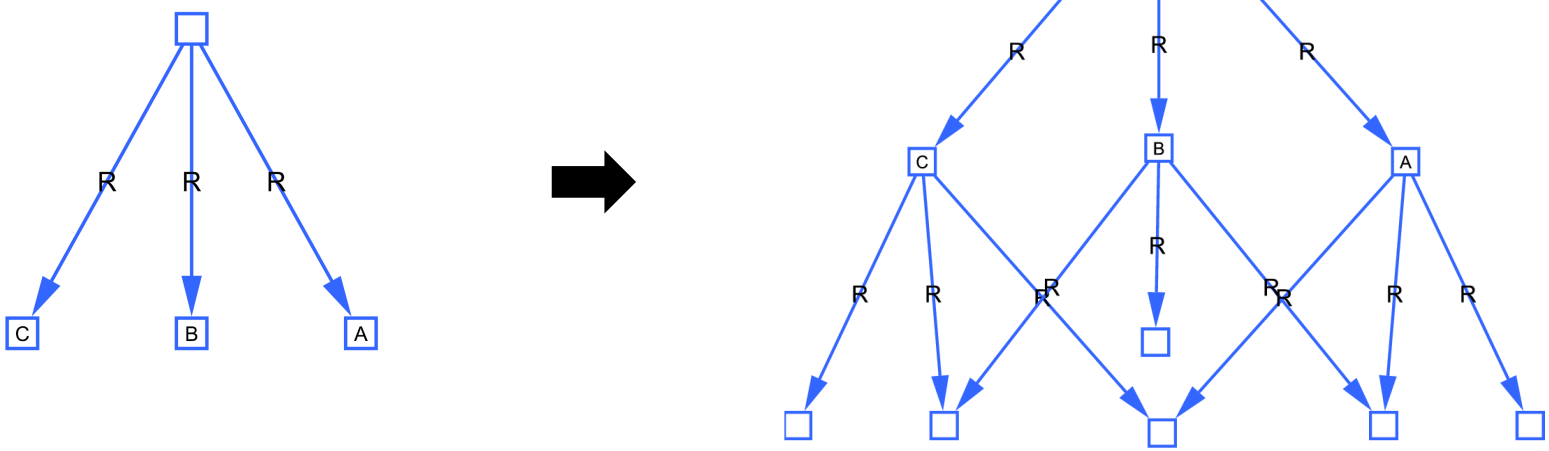
- Every successful pattern is considered
- Only unfruitful events are deleted

Related works

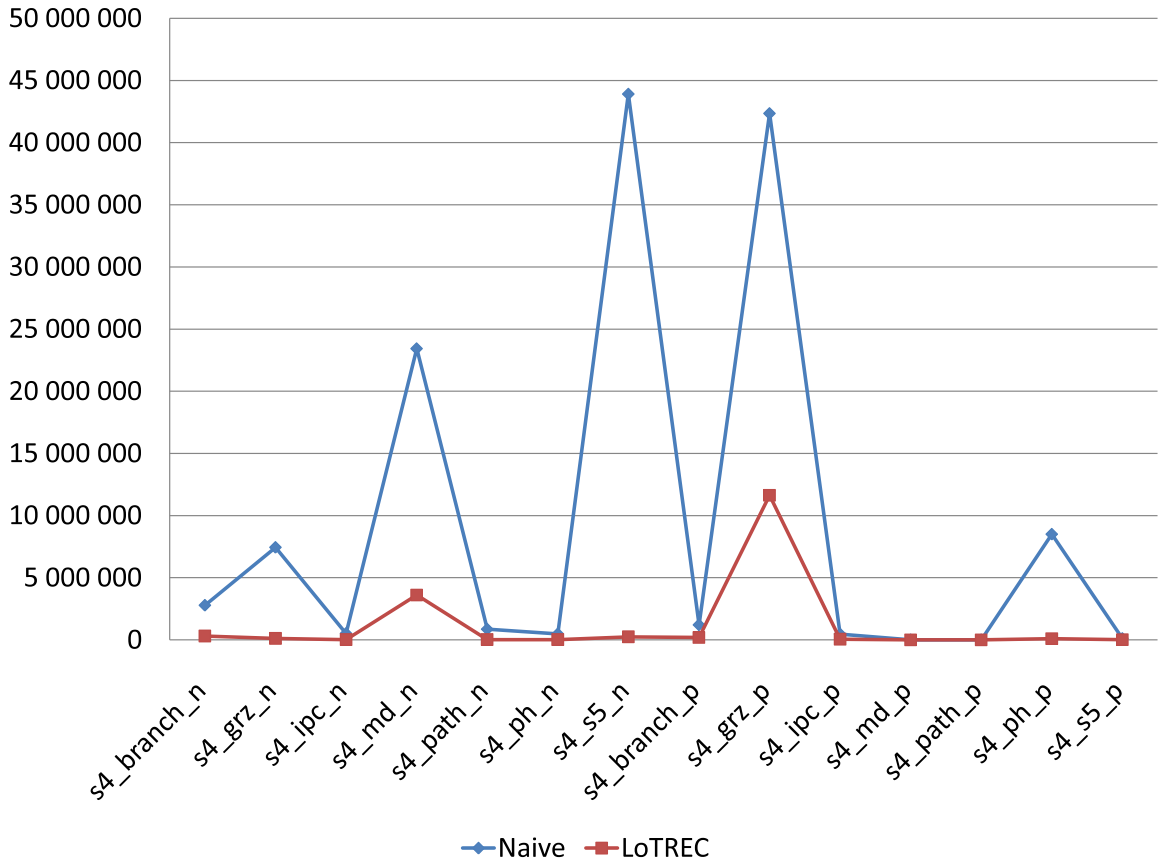
- PROGRES [Zündorf 99]
  - chooses an optimal plan over $|L|!$ local search plans
  - tracks invalid patterns
- Incremental Update [Varró & Varró 04]
  - tracks successful patterns in DB
  - stores & updates are space & time consuming

VS. other rewriting tools

- Comparison is not fair: formula matching
- Benchmark is hard to setup
- General purpose tools are not competent:
  3 levels confluent graph takes:
  - $\approx$ 6 sec in AGG
  - $<$ 0.5 sec in LoTREC



VS. naive pattern matching



Nb tentatives of pattern matching
Hardest S4 formulas in LWB benchmark

Conclusion

Research

Implementation of

- Time Sub-Intervals Logic
  [Goranko et al. 08]
- Public Announcement Logic (PAL)
  [De Lima et al. 09]

Not possible using another platform!

Academic

Accessed through logic courses:

- *Automated Reasoning*
  Prof. C. Pêcheur
  Université Catholique de Louvain, Belgium

- *FGI 3 - Logik*
  Dr. C. Eschenbach
  University of Hamburg, Germany

- *Logique, informatique et sciences cognitives*
  Prof. R. Villemaire
  University of Quebec at Montreal, Canada

Recap

Recap on the contributions of my thesis:

- Develop & maintain the LoTREC platform
- Study & implement new logics
- Promote the software in research & academic fields
- Establish the links with graph rewriting theory
- Study the event-based pattern matching
- Clarify the semantics of our rewriting system

2010: The Odyssey continues...

Currently:

- Book: "Kripke's World"
  Authors: [Gasquet, Herzig, Said, Schwarzentruber]

Next events:

- Universal Logic 2010, April - Lisbon (tutorial)
- ESSLLI 2010, August - Copenhagen, (1 week course)

What about?

- Language extension: SQL-queries?
- Performance: backtracking?
- Generic interface with other tools: will be user-friendly?
- Converse: what about CPDL?
- New methods for new logics...

Merci!