Calculating τ-Confluence Compositionally

Gordon J. Pace

University of Malta, Malta

Frédéric Lang, Radu Mateescu

INRIA Rhône-Alpes, France

Context

- Explicit state model-checking, state explosion…
- Compositional & on the fly verification
  - Intermediate model representation as network of LTSs (*composition expression*)
  - Local generation of LTS guided by verification needs
- Usually interested in properties up to branching bisimulation
  - Not all interleavings involving silent (τ) transitions are relevant

This talk

- Reduction techniques to eliminate irrelevant interleavings involving τ transitions
  - Based on strong τ-confluence (Groote & Selink 1996) and τ-prioritisation (Groote & van de Pol 2000)
  - *On the fly*
  - Using analysis of the composition expression architecture to eliminate τ transitions efficiently
  - Implemented in the CADP toolbox
- Techniques related to "partial order" reduction

  … but preserving branching bisimulation

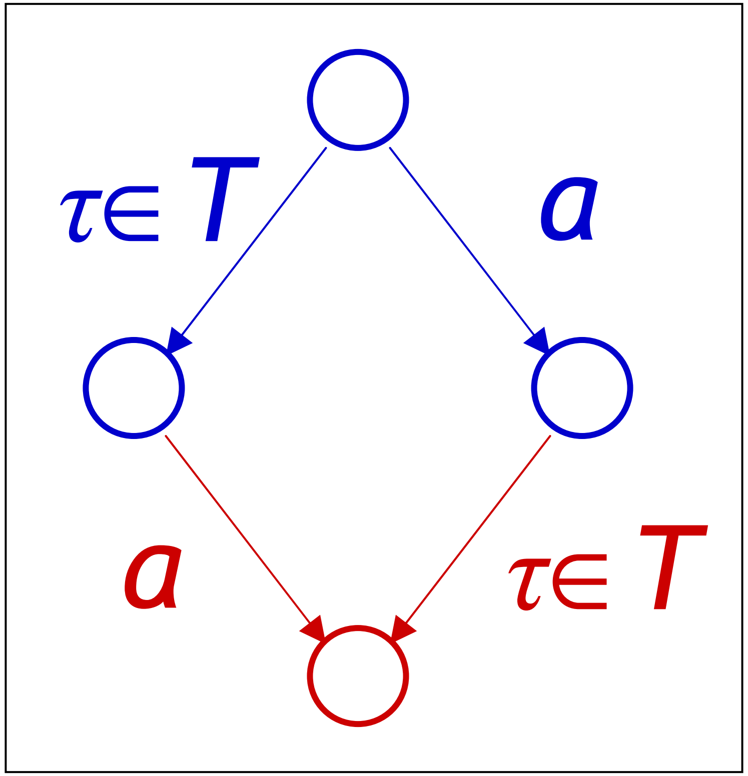
Strong τ-Confluence Intuition

A set of τ transitions *T* is *τ-confluent* if the system has the same behaviour *after* firing any transition in *T* as it had *before*

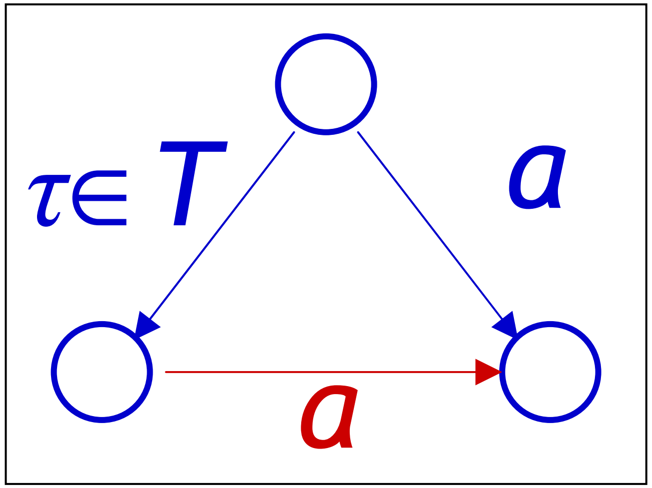
Strong τ-Confluence Definition

Blue arcs: for all

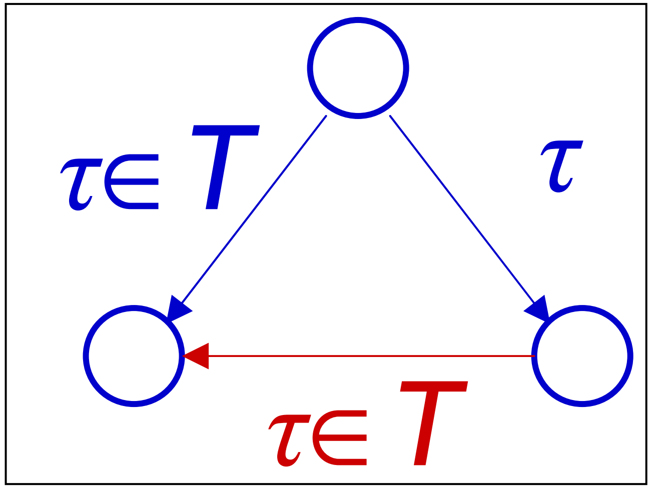
Red arcs: there exists



or



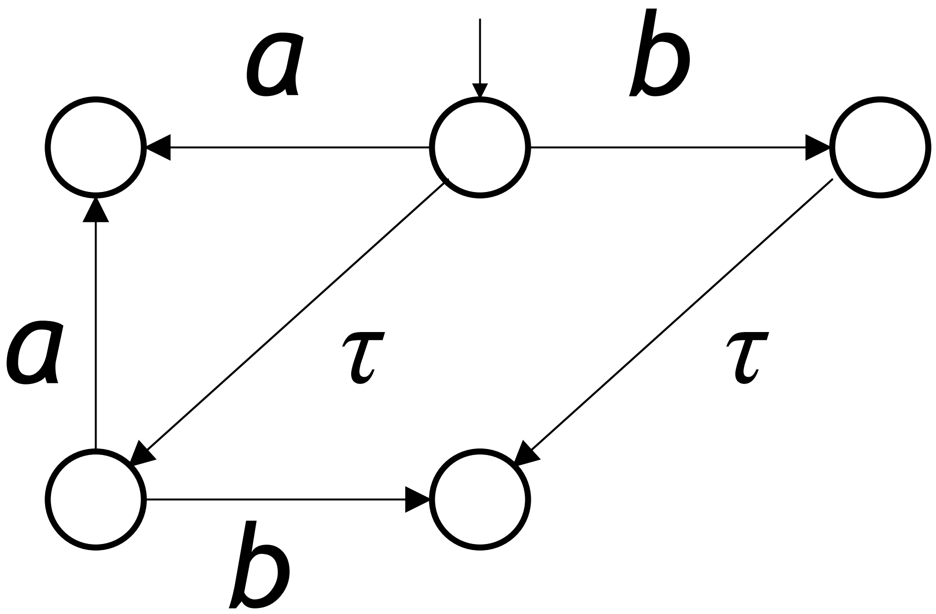
or



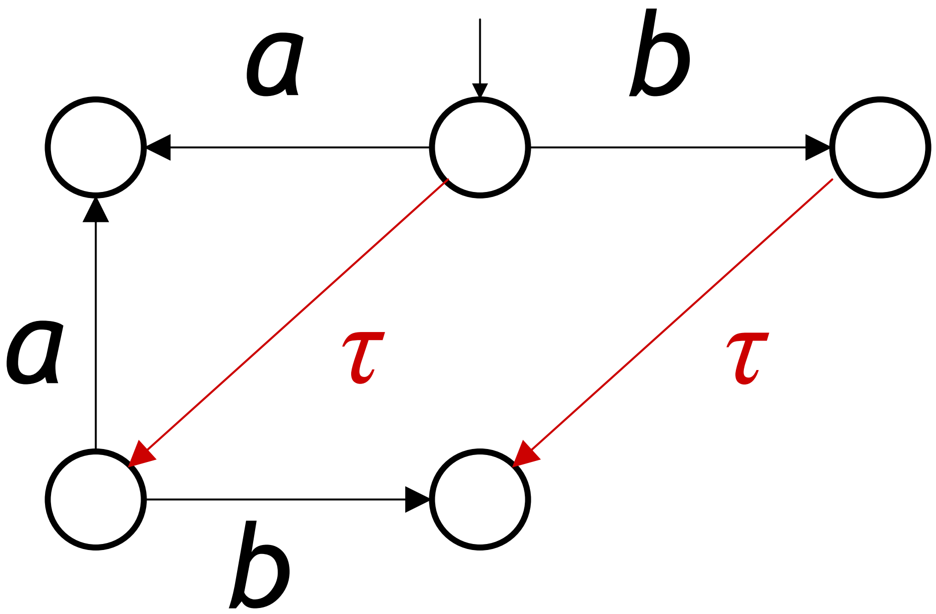
τ-Prioritisation Intuition

By removing any transition in choice
with a τ-confluent transition
the LTS remains unchanged
modulo branching bisimulation

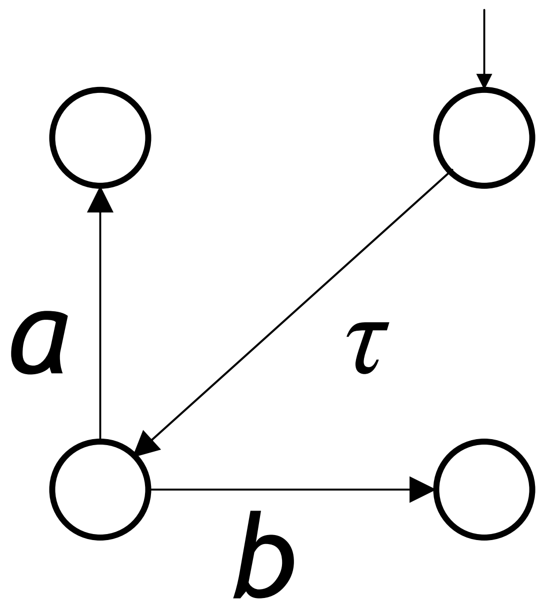
τ-Prioritisation Example



τ-Prioritisation Example

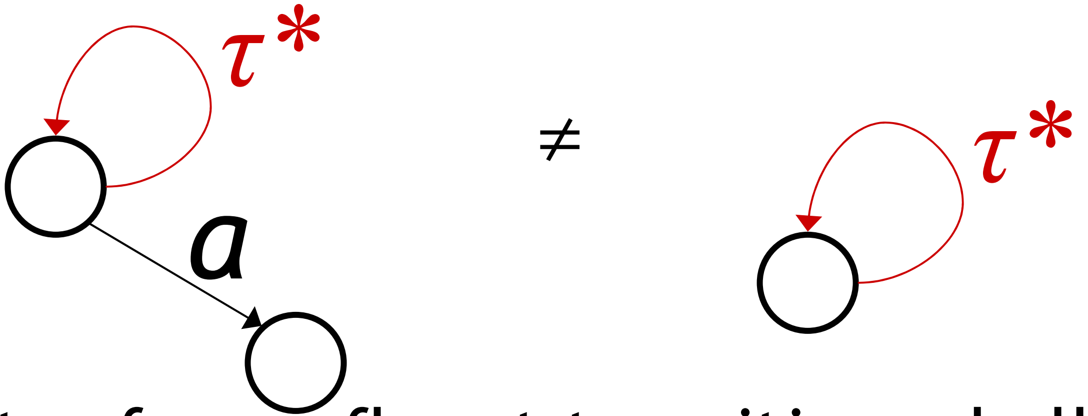


τ-Prioritisation Example

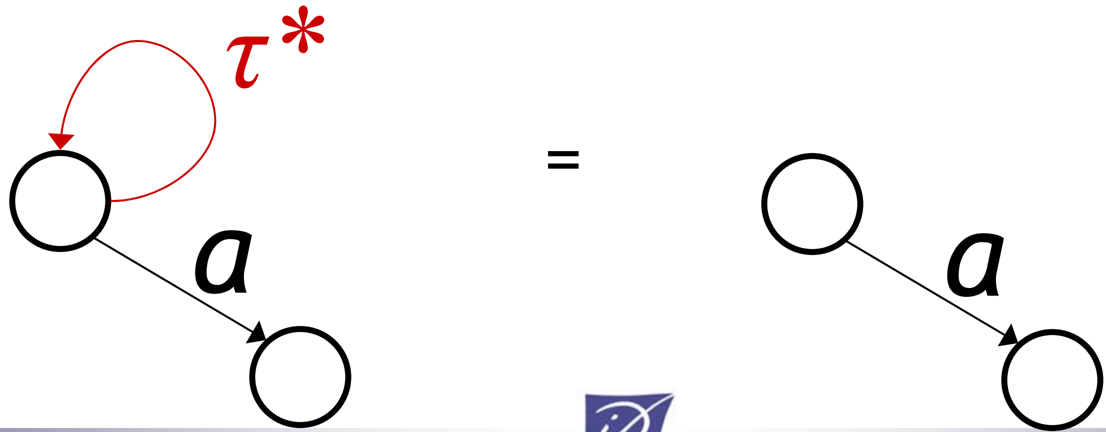


τ-Prioritisation and τ-Circuits

Exception: Circuit of τ-confluent transitions



Circuits of τ-confluent transitions shall be eliminated on the fly



Finding τ-Confluence

- Groote & van de Pol, MFCS 2000

  Global algorithm with complexity $O(m \times fanout_\tau^3)$ where

  - *m* is the total number of transitions in the LTS
  - $fanout_\tau$ is the maximal number of τ transitions in choice

- Blom & van de Pol, CAV 2002

  Automated theorem prover used to deduce confluence from a symbolic intermediate level description

Our Contribution

- Finding τ-confluence *on the fly* using Boolean Equation Systems

- Deducing τ-confluence in a system from that found in its (parallel) components

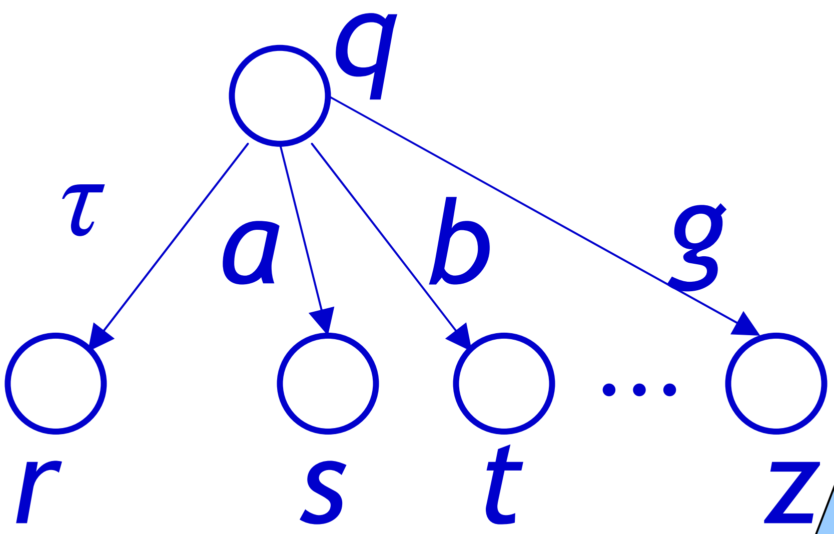
Boolean Equation Systems

Boolean Equation Systems (BESs) are made of

- A set of variables V
- For each variable v, an equation of the form $v = v_1 \vee \ldots \vee v_n$ or $v = v_1 \wedge \ldots \wedge v_n$

The least and greatest solution of a BES can be efficiently found with an on the fly algorithm (CAESAR_SOLVE library in CADP)

τ-Confluence Using BESs

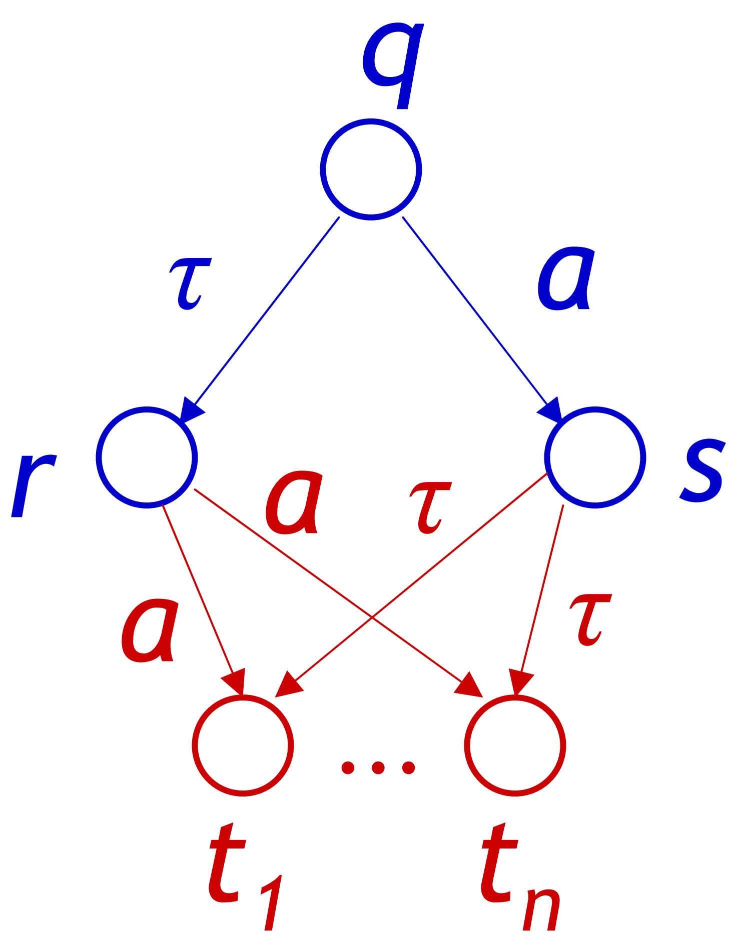


$$c_{q,r} = d_{q,r,s,a} \wedge \ldots \wedge d_{q,r,z,g}$$

The silent transition between *q* and *r* is confluent

The three states *q*, *r* and *s* can be closed in a τ-confluence diamond

Finding τ-Confluence Using BESs



$$d_{q,r,s,a} = c_{s,t1} \vee \ldots \vee c_{s,tn}$$

Finding τ-Confluence Using BESs

- Resolution procedure permits to find all τ-confluent transitions
- With complexity *O(m_τ x $fanout_\tau$ x fanout)* where
  - *m_τ* is the number of τ transitions in the LTS
  - *$fanout_\tau$* is the maximal number of τ transitions simultaneously fireable
  - *fanout* is the maximal number of transitions simultaneously fireable

Composition Expressions

Composition expressions are networks of LTSs built upon LOTOS *parallel composition* and *hiding*

```
hide R_T1, R_T2, R1, R2 in
        CRASH_TRANSMITTER
        |[R_T1, R_T2]|
        (
                (RECEIVER_THREAD1 || FAIL_RECEIVER1)
                |[R1, R2]|
                (RECEIVER_THREAD2 || FAIL_RECEIVER2)
        )
```

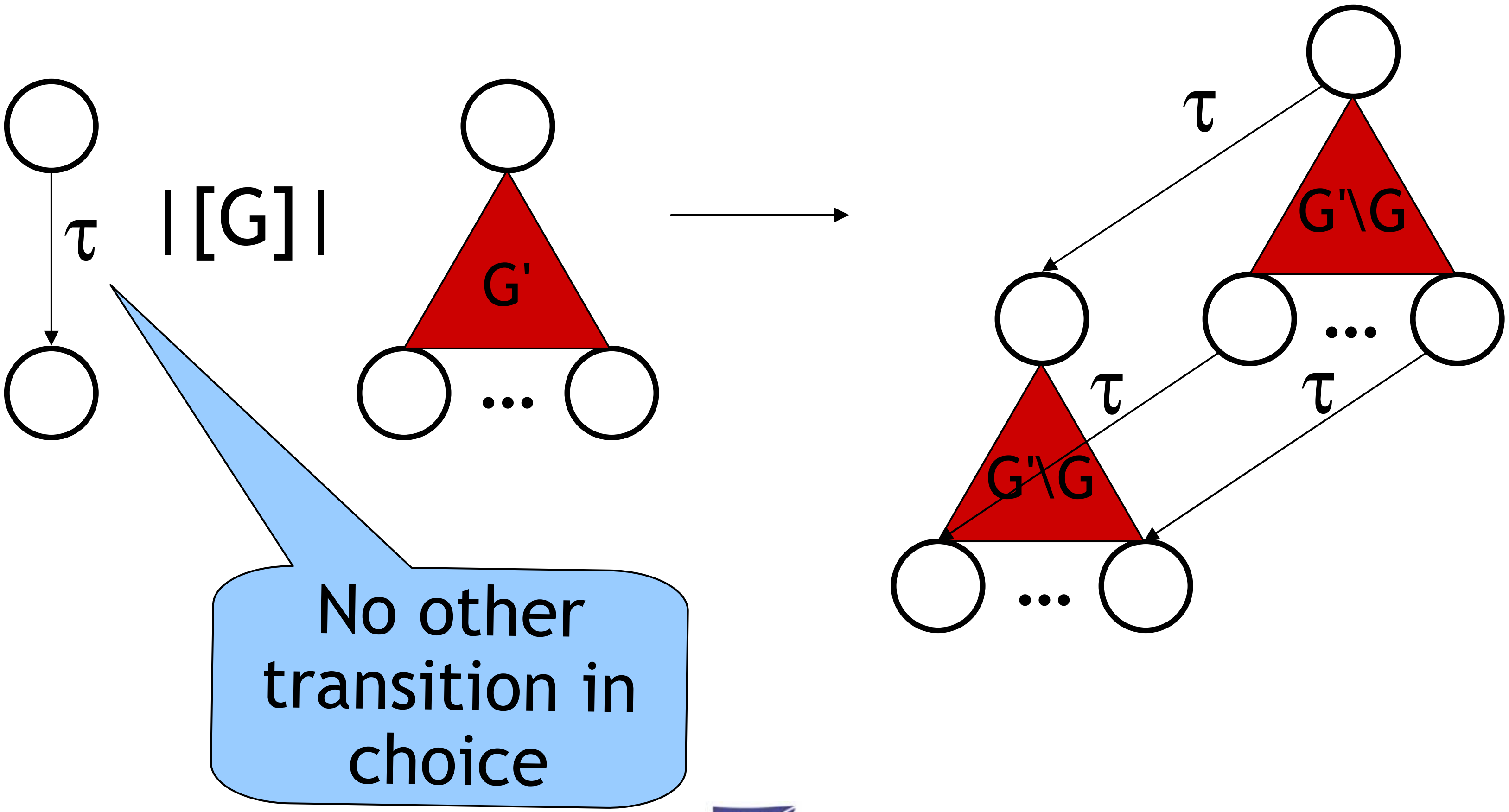
Finding τ-Confluence in Composition Expressions

Theorem 1: τ-confluent transitions in an LTS appearing in a composition expression generate only τ-confluent transitions

By calculating τ-confluent transitions of (small) components, some τ-confluence in the resulting compound LTS can be identified

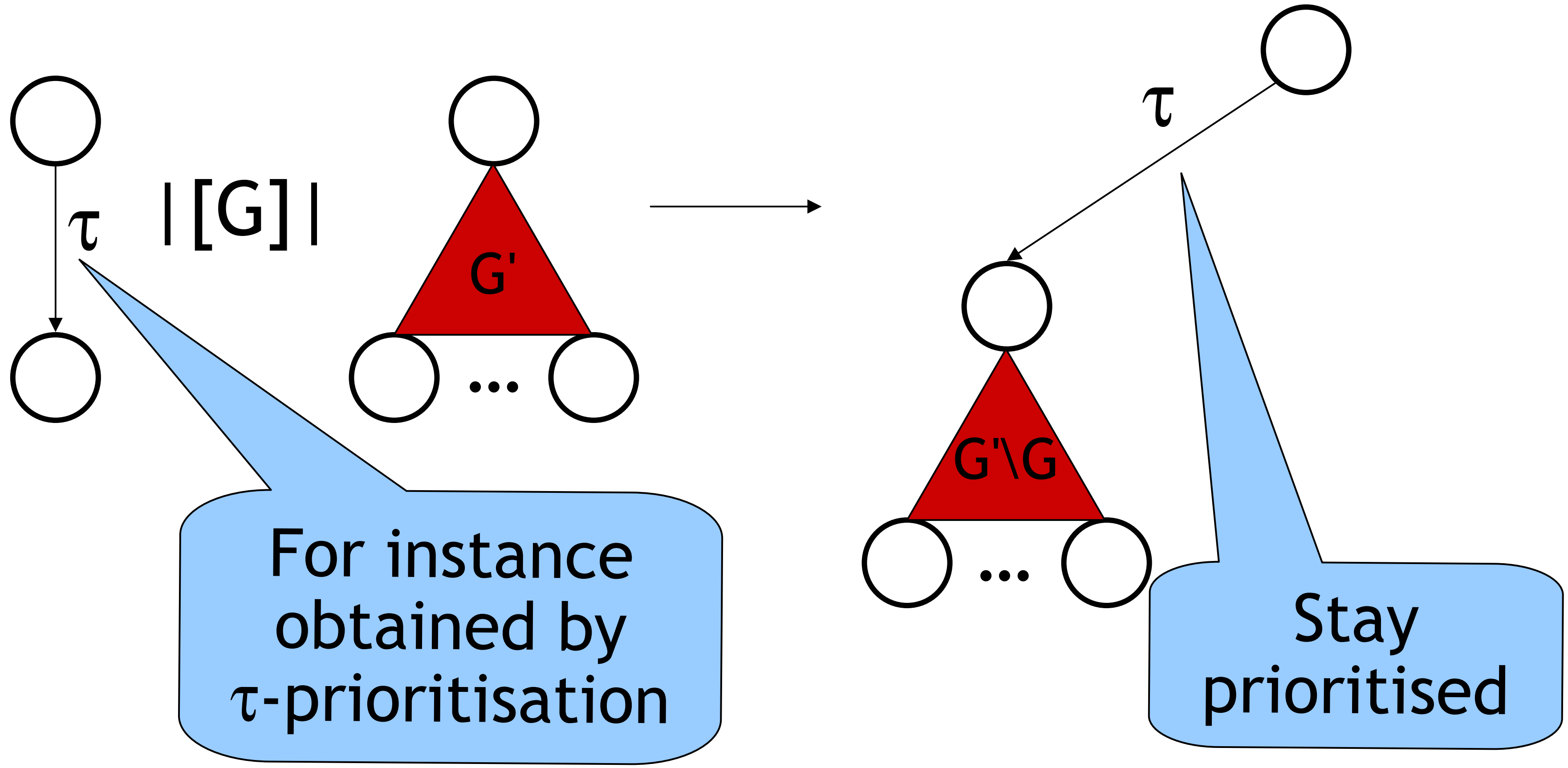
τ-Confluence & Composition

Particular case of Theorem 1



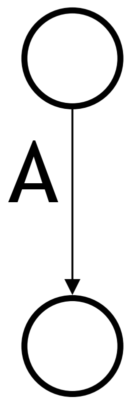
τ-Confluence & Composition

Particular case of Theorem 1



τ-Confluence & Composition

There are also locally visible transitions that may lead to τ-confluent transitions



can be prioritised if

(1) A is hidden in the context of the expression

(2) A is not synchronised in the context

(3) there is no other transition locally in choice with A

Finding τ-Confluence in Composition Expressions

Theorem 2: A conservative set of transitions *P* can be identified such that only the transitions generated by *P* have a chance to be confluent

By calculating *P*, we can assume that any transitions not generated by *P* are not τ-confluent in the resulting compound LTS

Finding τ-Confluence in Composition Expressions

- Theorems 1 & 2 can be used to partially *deduce* τ-confluence without the need to apply the BES algorithm globally
- Tools implemented in CADP
  - τ-CONFLUENCE: BES based algorithm
  - EXP.OPEN 2.0: Compositional τ-confluence deduction (Theorem 1)

Experiment: rel/REL

Reliable atomic multicast protocol between one transmitter and several receivers

```
hide R_T1, R_T2, R1, R2 in
    CRASH_TRANSMITTER
    |[R_T1, R_T2]|
    (
        (RECEIVER_THREAD1 || FAIL_RECEIVER1)
        |[R1, R2]|
        (RECEIVER_THREAD2 || FAIL_RECEIVER2)
    )
```

Experiment: rel/REL

Normal generation versus on the fly τ-prioritisation of processes

	Normal		τ -prioritised		Difference %	
	states	transitions	states	transitions	states	transitions
CRASH_TRANSMITTER	85	108	73	84	14%	22%
RECEIVER_THREAD*n*	16 260	167 829	16 260	115 697	0%	31%
FAIL_RECEIVER*n*	130	1 059	130	1 059	0%	0%

Experiment: rel/REL

Cost and effect of τ-prioritisation in composition expression

	Normal	τ-prioritised	Difference %
Number of states	249 357	114 621	54%
Number of transitions	783 470	220 754	72%
EXP.OPEN execution time	2m23s	2m10s	9%
EXP.OPEN memory consumption (Kb)	5 776	3 944	32%
SVL execution time	3m05s	3m03s	1%

Conclusions

- Efficient techniques on selected examples
  - τ-confluence is created mostly by parallel composition
  - But the memory overhead is negligible in worst cases
- On the fly τ-prioritisation can be used as preprocessing step for branching minimisation
- Results are not limited to LOTOS-like expressions

  EXP.OPEN implements other operators (CCS, CSP, muCRL, E-LOTOS) using synchronization vectors
- Potential τ-confluence still to be exploited in tools
- CADP web page: http://www.inrialpes.fr/vasy/cadp