On Negotiation as Concurrency Primitive

Javier Esparza, Techn. Univ. München (D)

Jörg Desel, FernUniversität in Hagen (D)

On Negotiation as Concurrency Primitive I:

arbitrary / **weakly deterministic** /deterministic
cyclic / **acyclic** Negotiations

[CONCUR 2013]

Javier Esparza, Techn. Univ. München (D)

Jörg Desel, FernUniversität in Hagen (D)

On Negotiation as Concurrency Primitive **II**:

arbitrary / weakly deterministic /**deterministic**
cyclic / acyclic Negotiations

[FOSSACS 2014]

Javier Esparza, Techn. Univ. München (D)

Jörg Desel, FernUniversität in Hagen (D)

Negotiation Programs

[unpublished]

Javier Esparza, Techn. Univ. München (D)

Jörg Desel, FernUniversität in Hagen (D)

(Multiparty) negotiation

Negotiation as a metaphor for distributed problem solving

R Davis, RG Smith - Artificial intelligence, 1983 - Elsevier

Automated negotiation: prospects, methods and challenges

soton.ac.

NR Jennings, P Faratin, AR Lomuscio... - ... and **Negotiation**, 2001 - Springer

AUTOMATED NEGOTIATION: PROSPECTS, METHODS AND CHALLENGES Group Decision and **Negotiation** 10: 199–215, 2001 © 2001 Kluwer Academic Publishers. Printed in the Netherlands

[BOOK] Rules of encounter: designing conventions for **automated negotiation** among computers

JS Rosenschein - 1994 - books.google.com

Rules of Encounter applies the general approach and the mathematical tools of game theory in a formal analysis of rules (or protocols) governing the high-level behavior of interacting heterogeneous computer systems. It describes a theory of high-level protocol design that ...

Cited by 1760 Related articles All 8 versions Import into BibTeX More

negotiation-based protocols, meta-data, information dissemination 1. Introduction ...

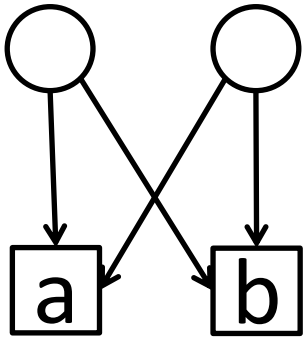
Cited by 1035 Related articles All 22 versions Import into BibTeX More

Negotiation as concurrency primitive

- Concurrency theory point of view:

Negotiation = synchronized choice

- CSP: $(aP_1 + bP_2) \,|\{a, b\}|\, (aQ_1 + bQ_2)$
- Petri nets:

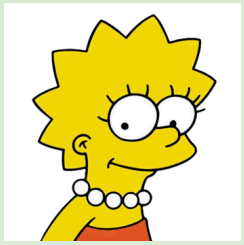


- Negotiations: a net-like concurrency model with negotiation as primitive.

The Father-Daughter-Mother Negotiation

Agents

Father

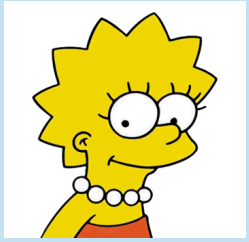


Daughter

Mother

Initial states

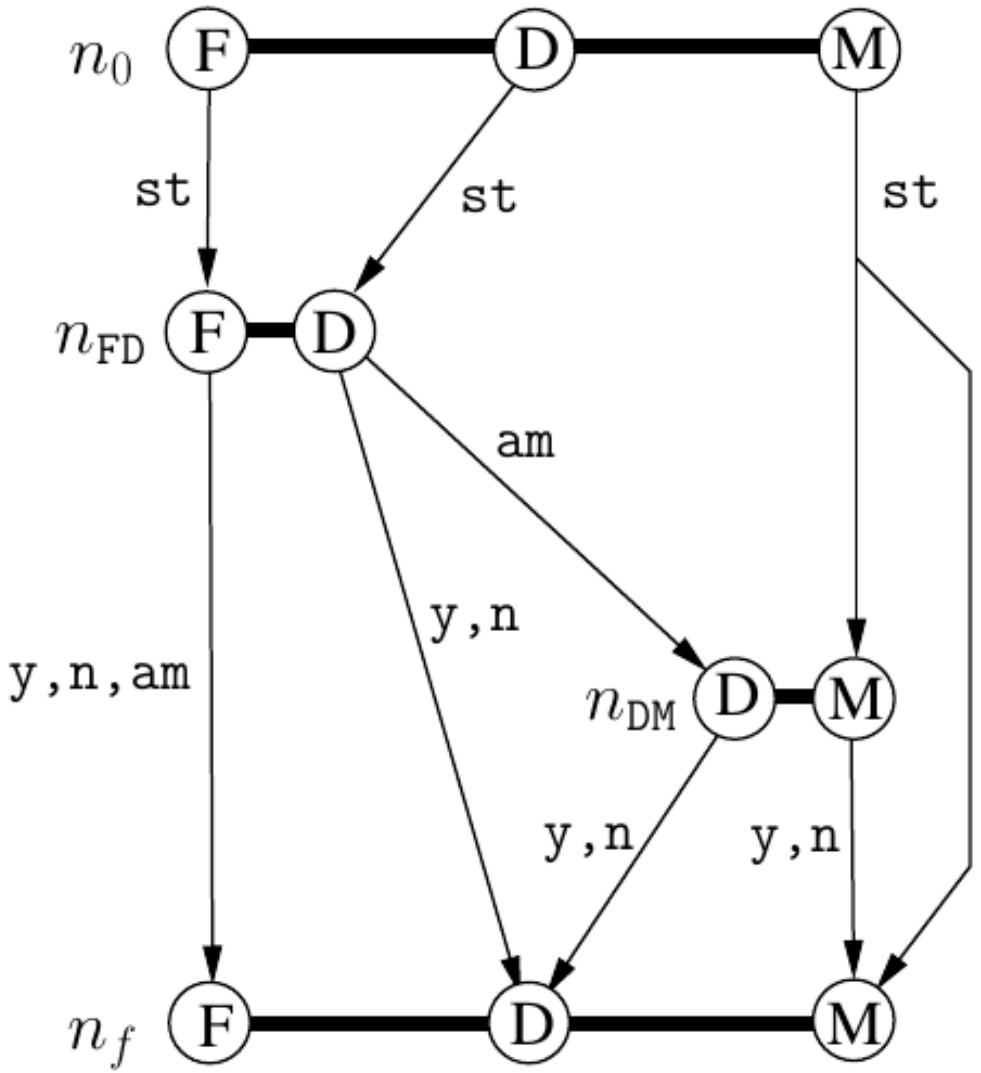
11:00 pm



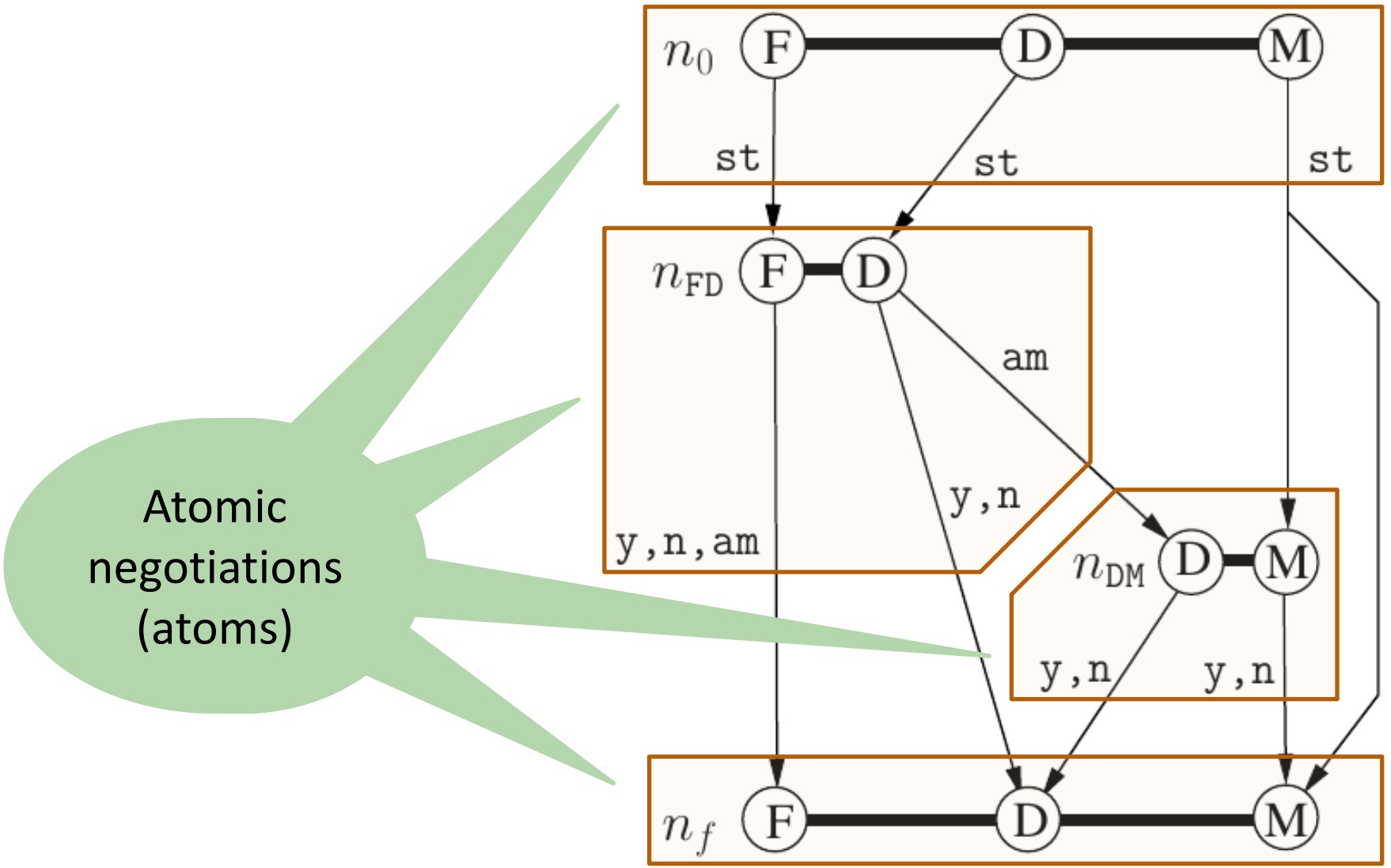
2:00 am

10:00 pm

The Father-Daughter-Mother Negotiation



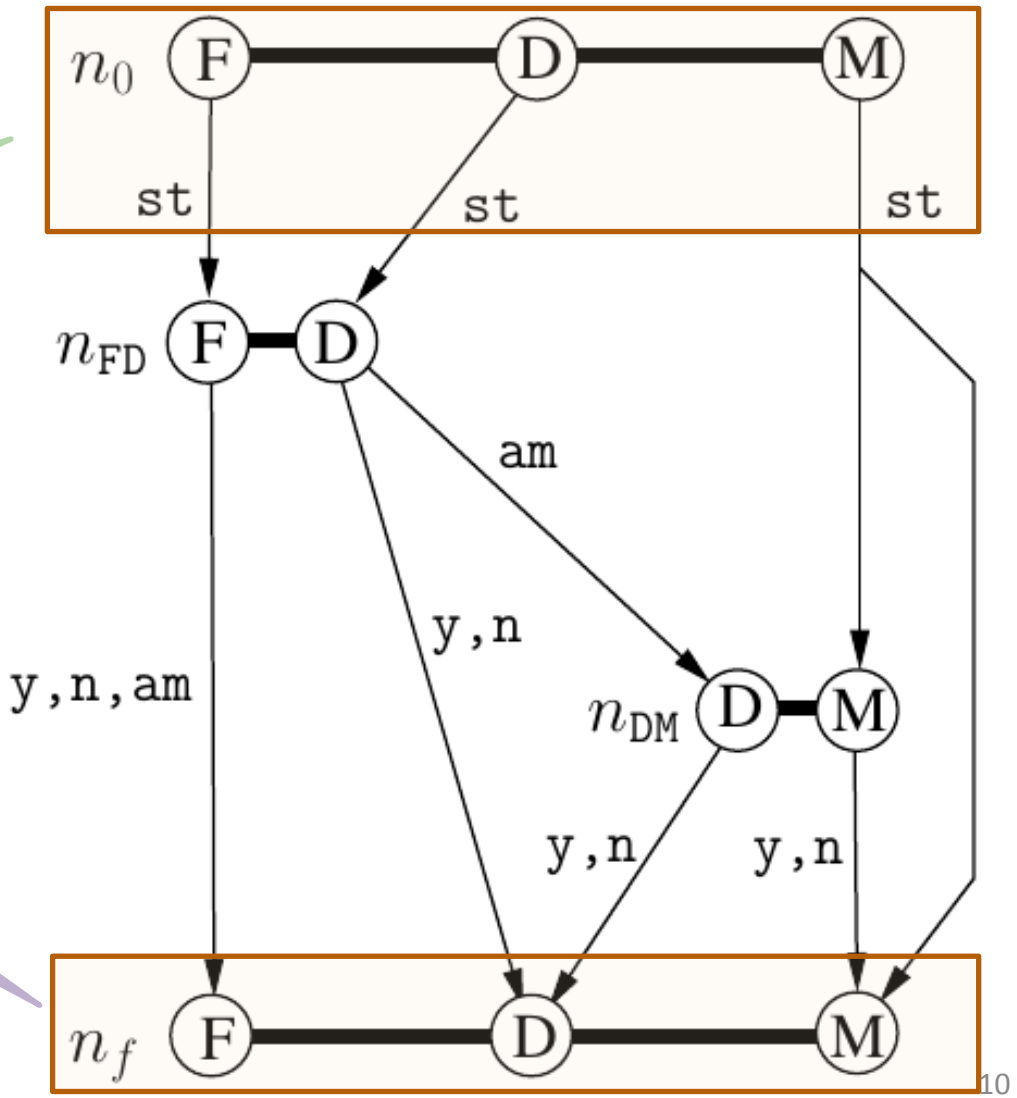
The Father-Daughter-Mother Negotiation



The Father-Daughter-Mother Negotiation

Initial atom

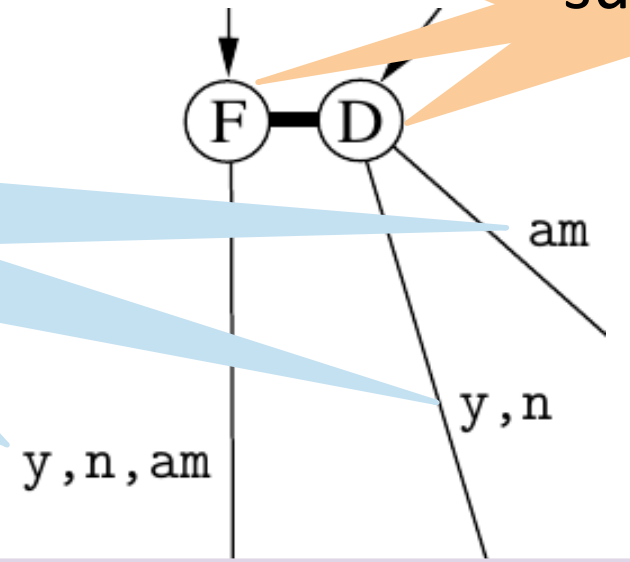
Final atom



An atomic negotiation

Parties:
subset of agents

Outcomes:
yes, no,
ask_mum



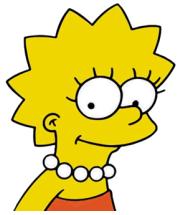
State transformers (one per outcome):

$$T_{am}(tf, td) = (tf, td)$$

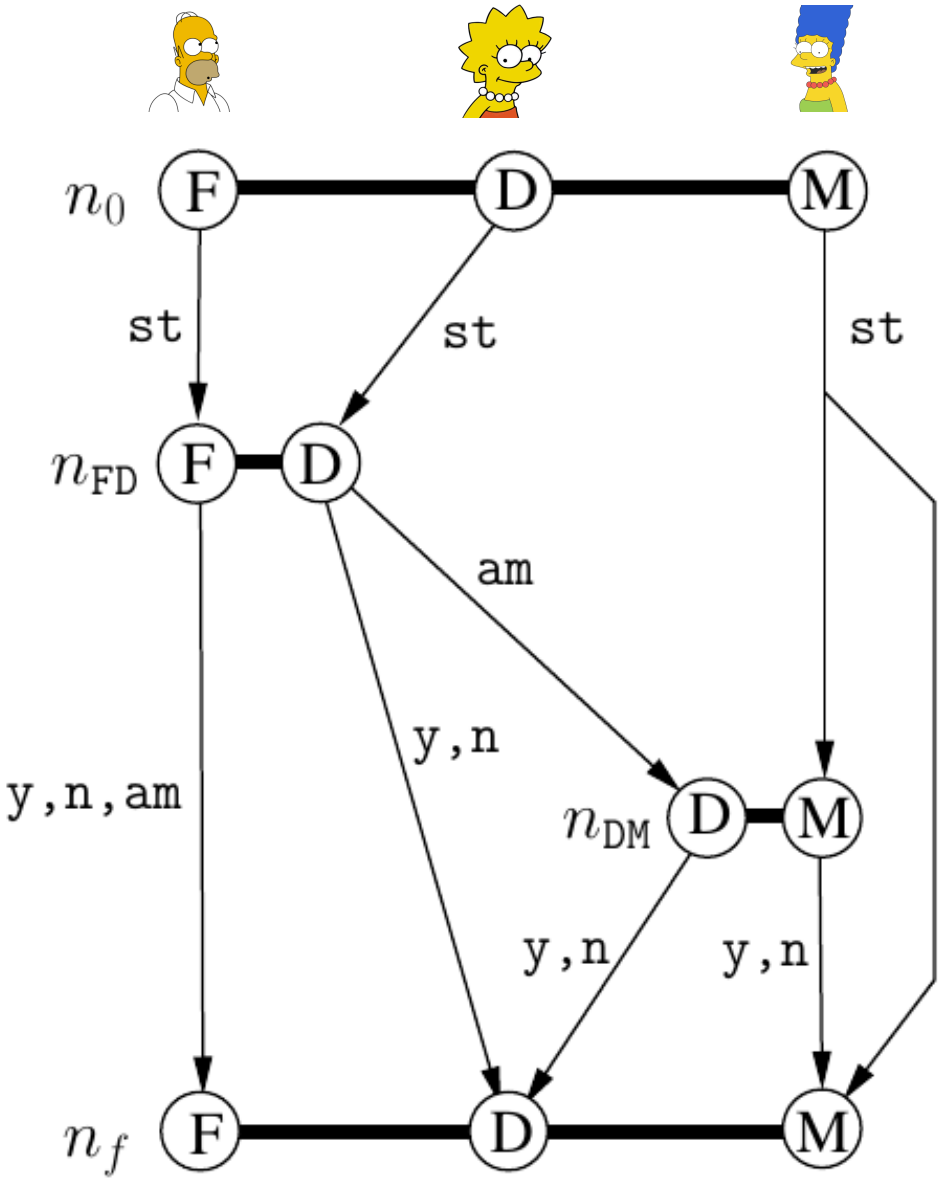
$$\mathrm{T}_y(tf, td) = \{(t, t) \mid tf \leq t \leq td\}$$

(sometimes we identify a and T_a)

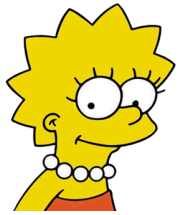
Semantics

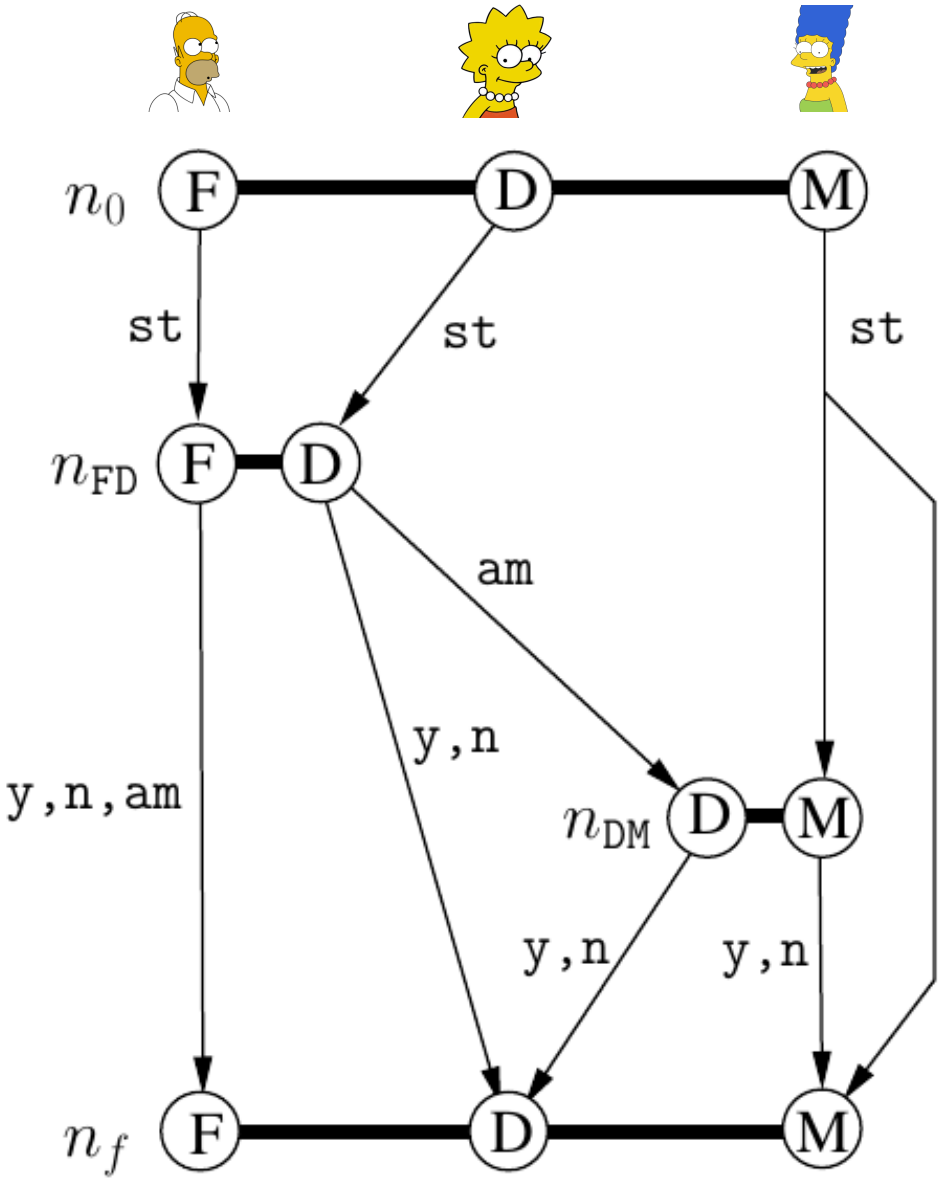

11:00	2:00	10:00
11:00	2:00	10:00
12:00	12:00	10:00
12:00	12:00	12:00



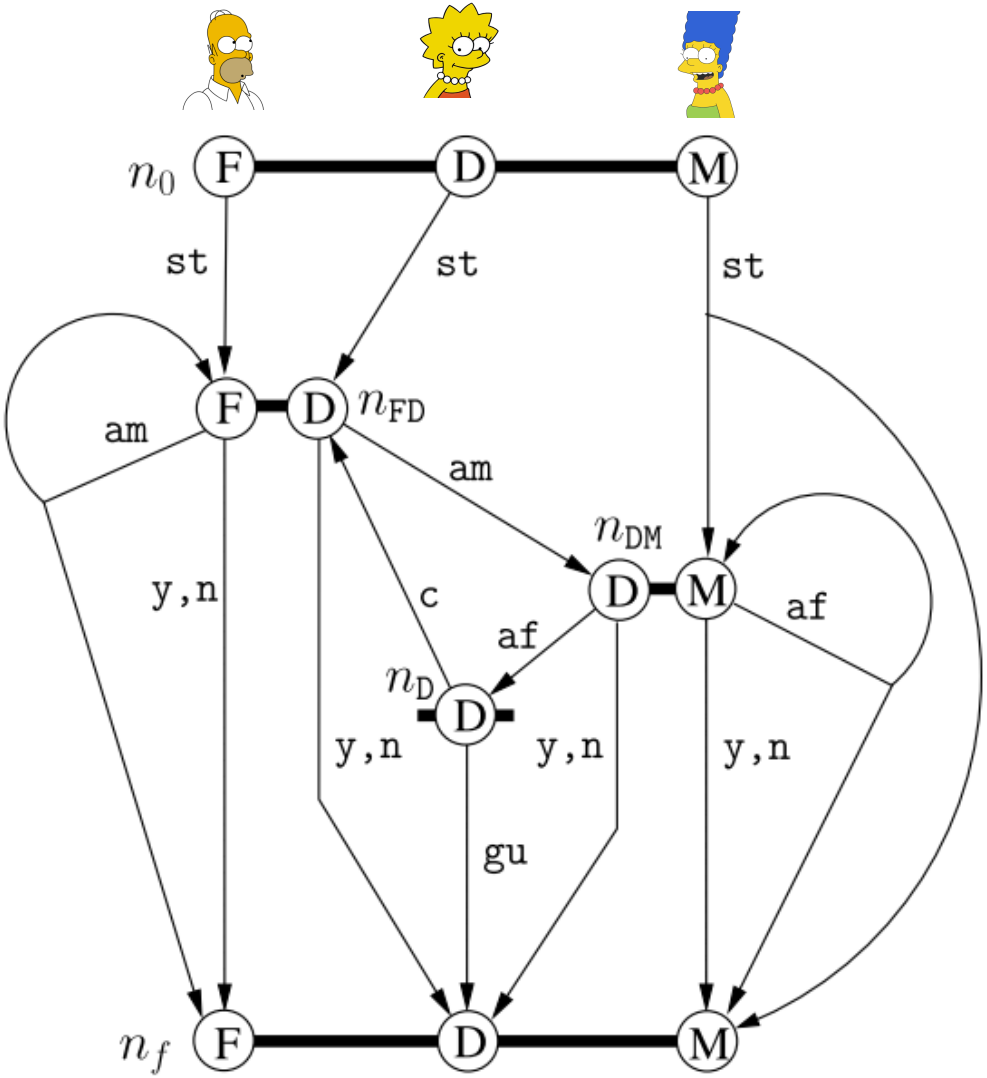
Semantics

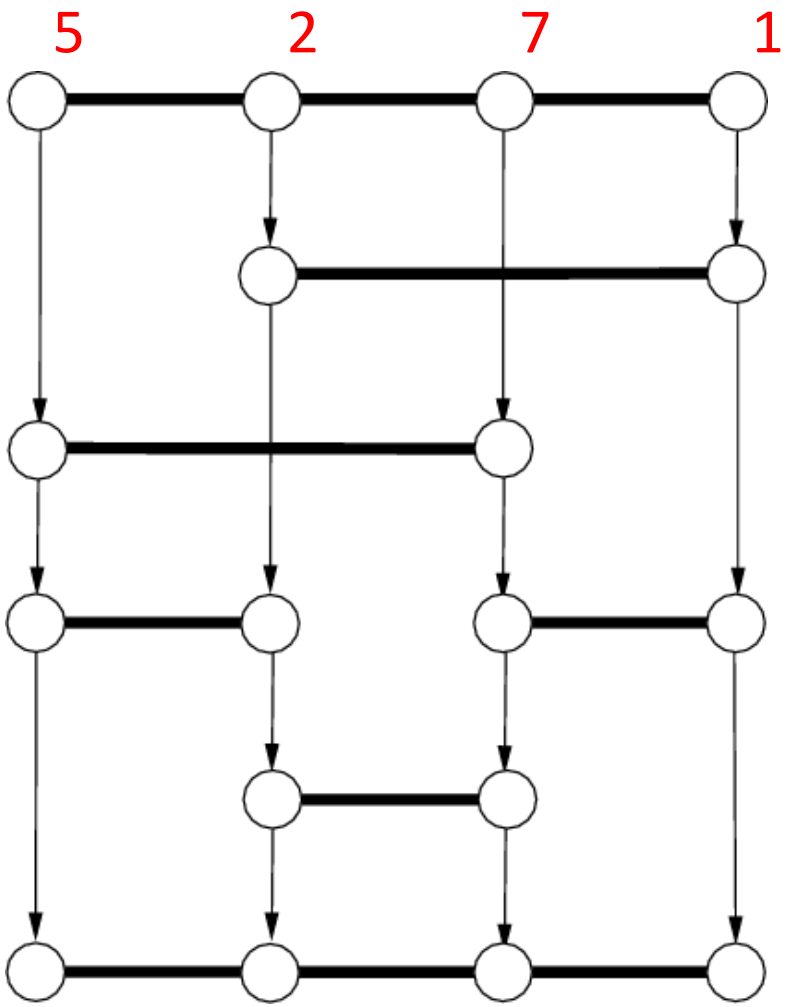

11:00	2:00	10:00
11:00	2:00	10:00
11:00	2:00	10:00
11:00	Angry!	Angry!
Beer!	Angry!	Angry!



The Ping-Pong Negotiation



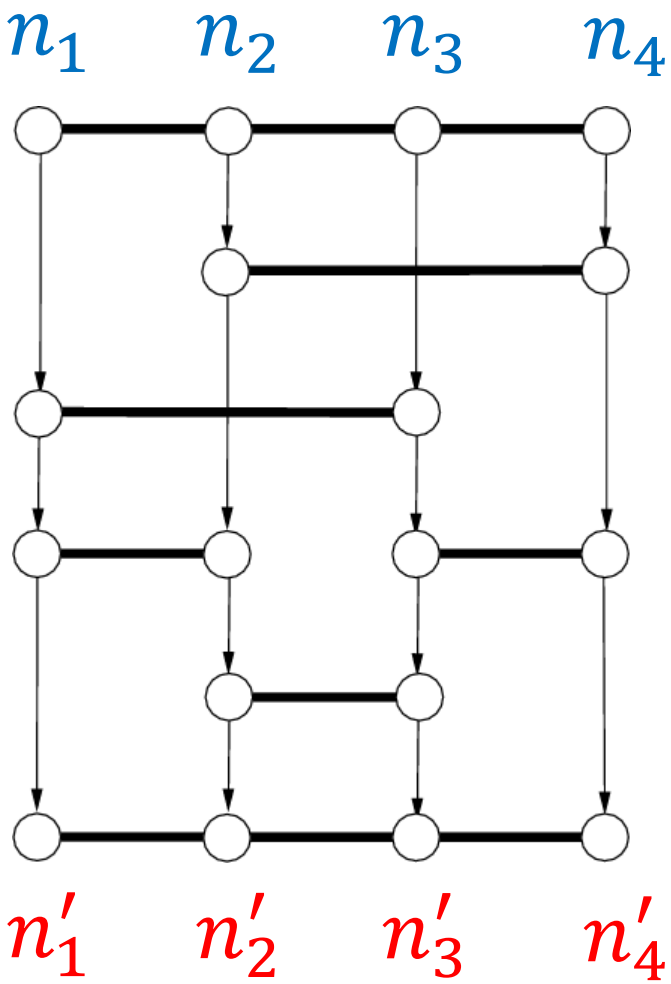
Negotiations as Parallel Computation Model. An Example: Sorting four integers



- Four agents.
- Internal states: integers.
- Transformer of internal atom: swap integers if not in ascending order.

$$T(x,y) = \text{if } y < x \text{ then } (y,x) \text{ else } (x,y)$$

Analysis of the sorting negotiation



Sorting negotiation correct if

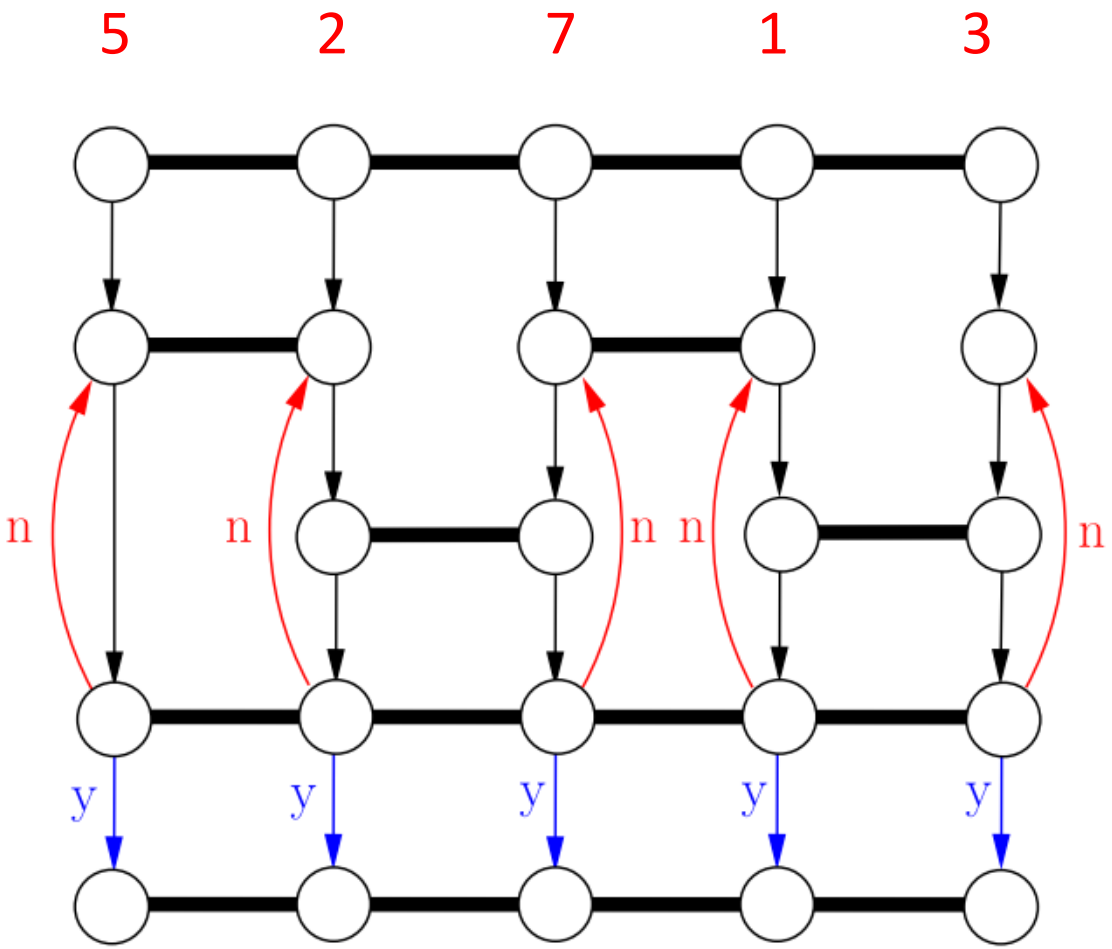
- sound, and
- summary consists of all pairs

((n_1, n_2, n_3, n_4) , (n'_1, n'_2, n'_3, n'_4))

s. t. $n'_1 n'_2 n'_3 n'_4$ is a permutation of

$n_1 n_2 n_3 n_4$ and $n'_1 \leq n'_2 \leq n'_3 \leq n'_4$

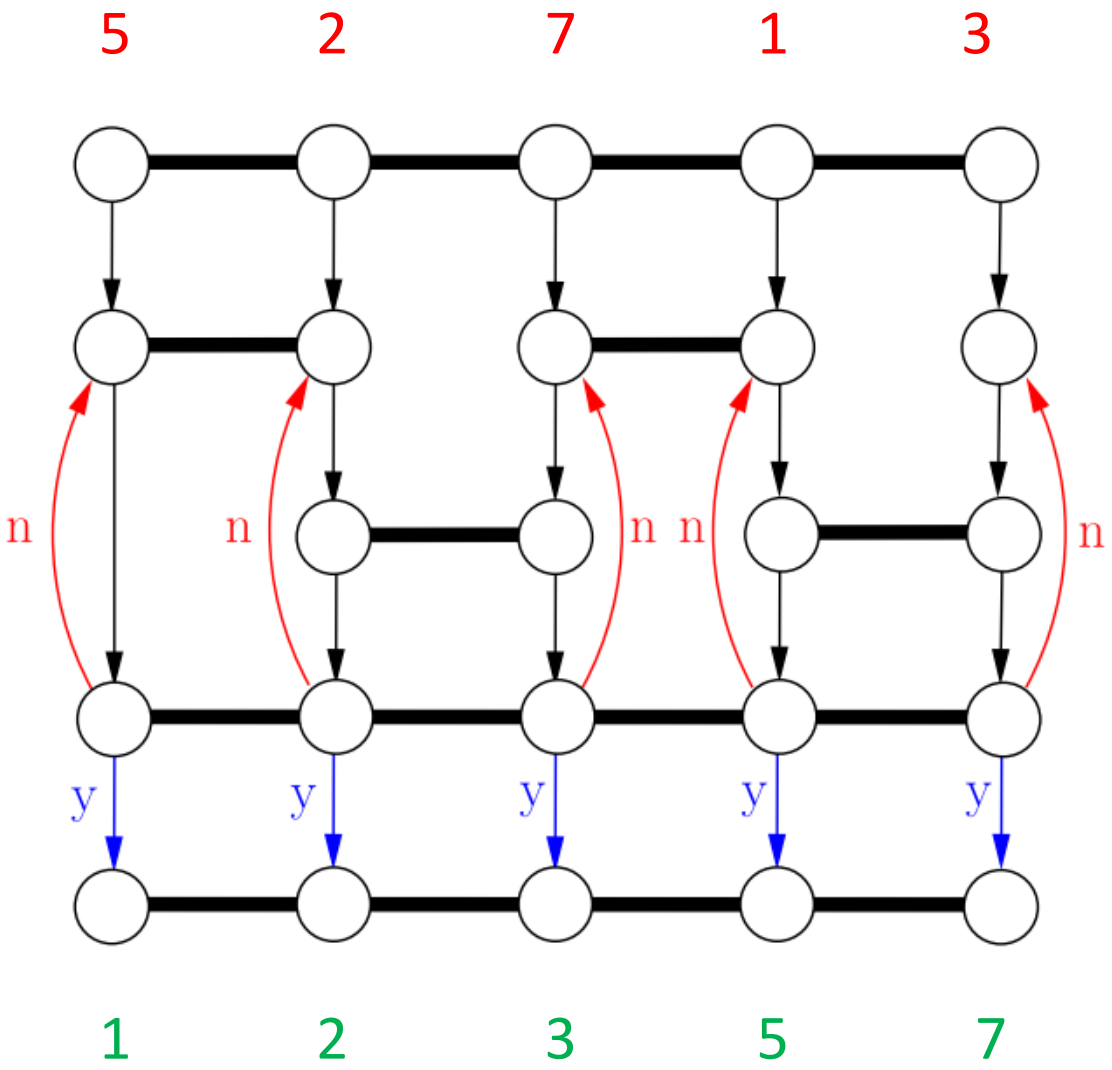
Negotiations as Parallel Computation Model. Parallel Bubblesort



- Five agents.
- Internal states: integers.
- Transformer of internal binary atom: swap integers if not in ascending order.

$$T(x,y) = \text{if } y < x \text{ then } (y,x) \text{ else } (x,y)$$

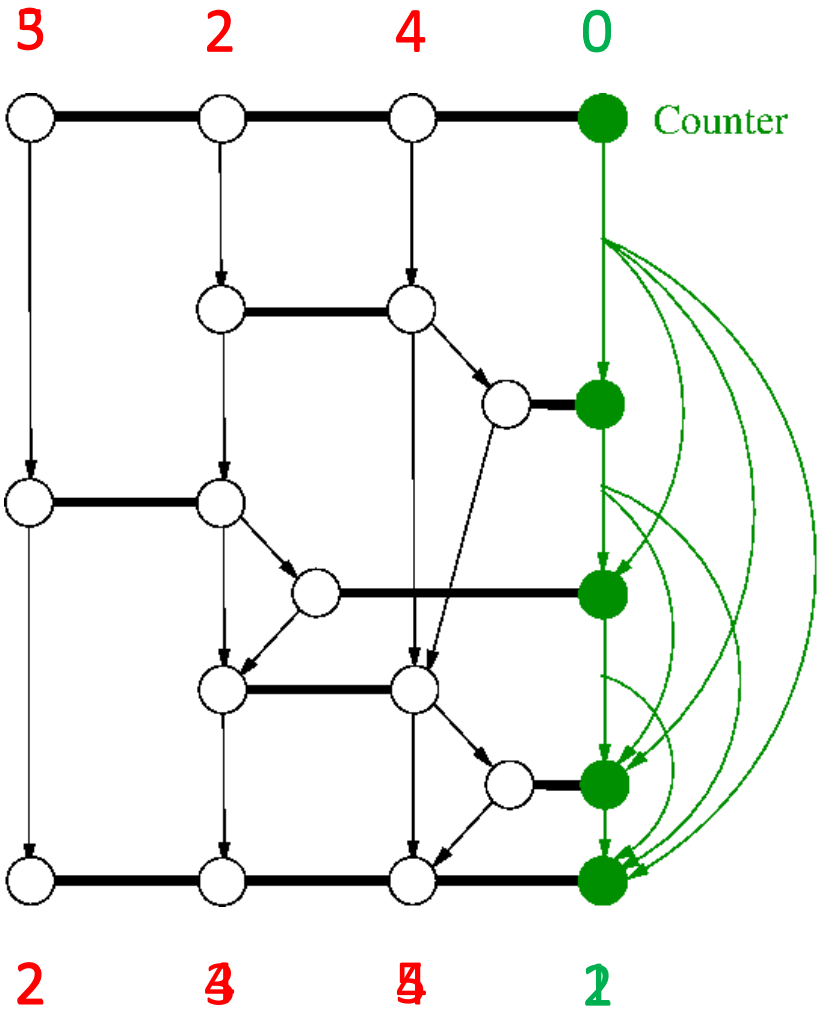
Negotiations as Parallel Computation Model. Parallel Bubblesort



- Five agents.
- Internal states: integers.
- Transformer of internal binary atom: swap integers if not in ascending order.

$$T(x,y) = \text{if } y < x \text{ then } (y,x) \text{ else } (x,y)$$

Counting pairs of consecutive numbers

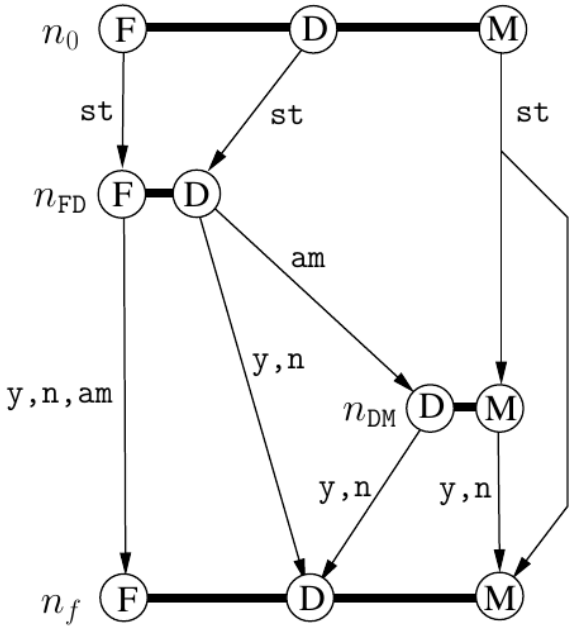


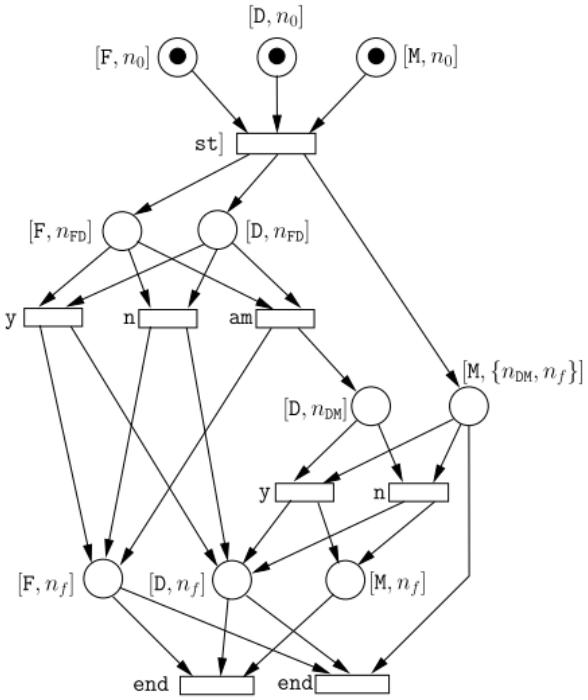
- Sort three integers and count the number of consecutive pairs
- Three agents communicate with a fourth Counter agent

- Is it correct …?

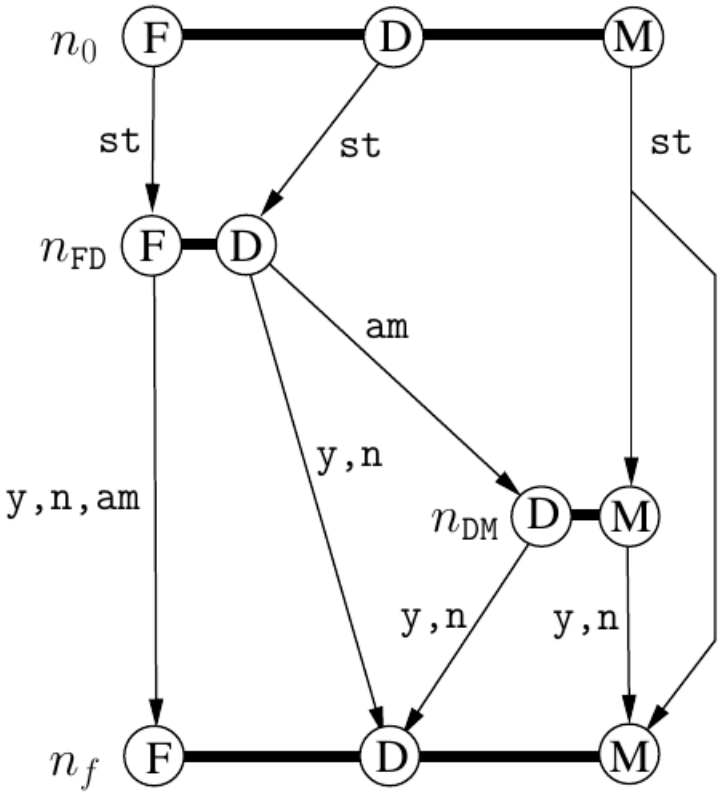
Negotiations and Petri nets

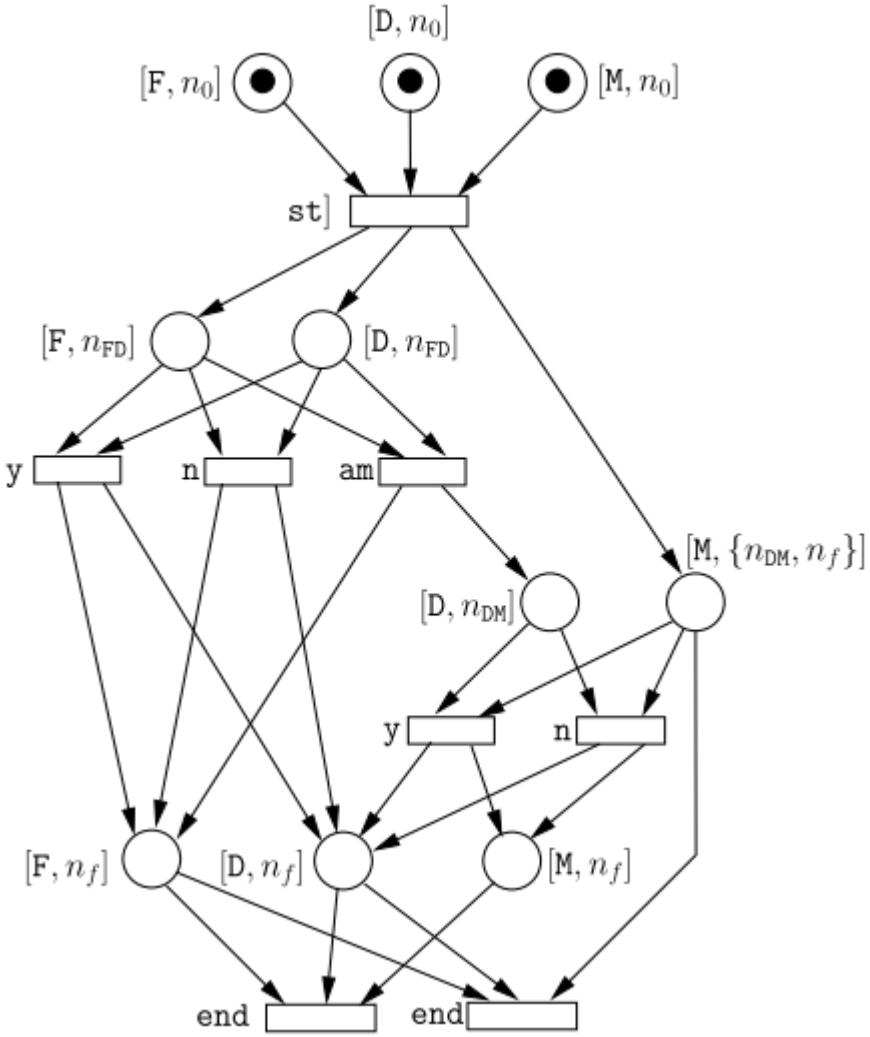
- Negotiations have the same expressive power as (coloured) 1-safe Petri nets.
- However, negotiations can be exponentially more succint.





Negotiations → 1-safe Petri nets





Negotiations → 1-safe Petri nets

$[D, n_{DM}]$

$[M, \{n_{DM}, n_f\}]$

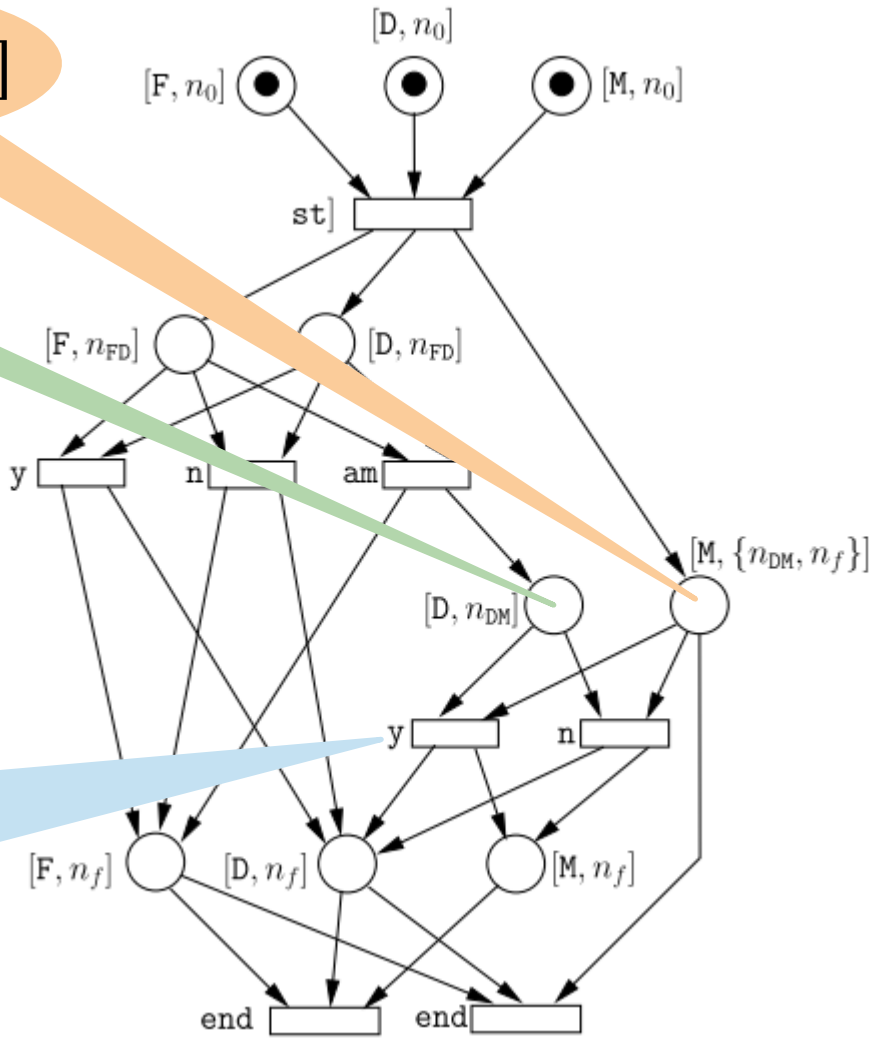
Meaning of a token in $[D, n_{DM}]$:

agent D is currently ready to engage in the atom n_{DM} (and no others)

Meaning of a token in $[M, \{n_{DM}, n_f\}]$:

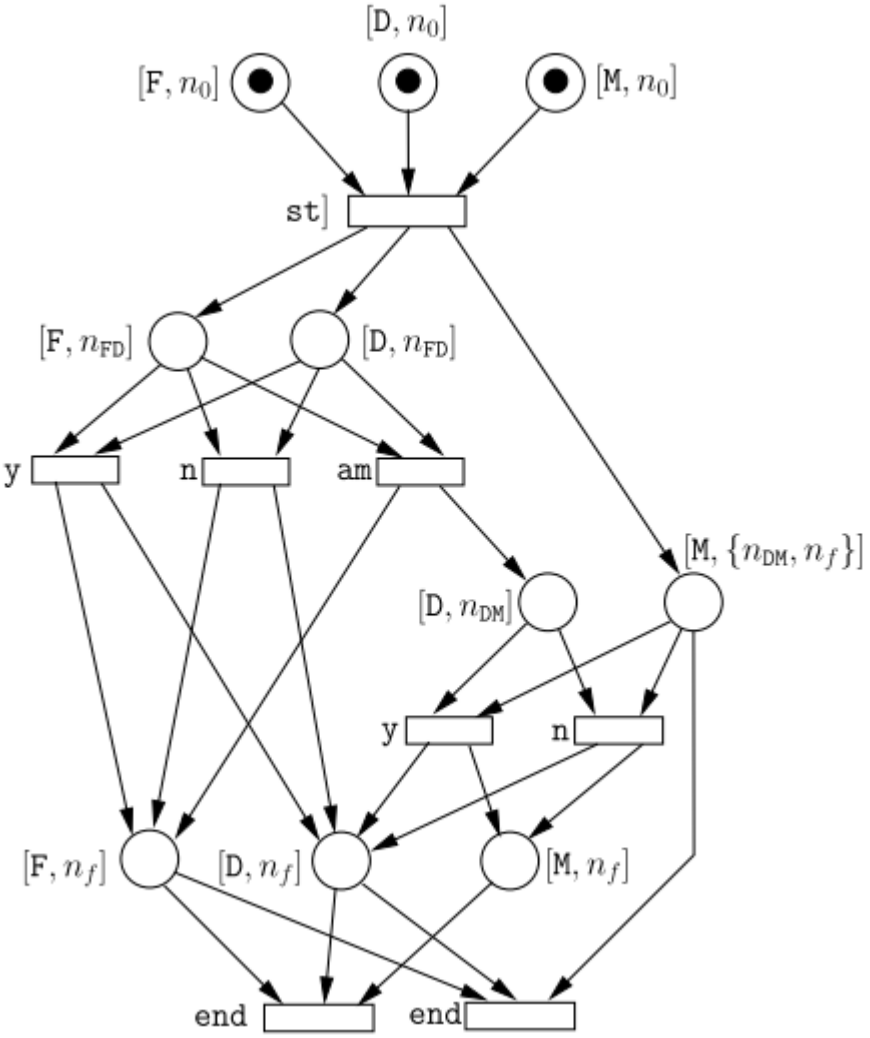
agent M is currently ready to engage in the atoms n_{DM} and n_f (and no others)

DM takes place with outcome „yes“, after which *D* and *M* are only ready to engage in n_f

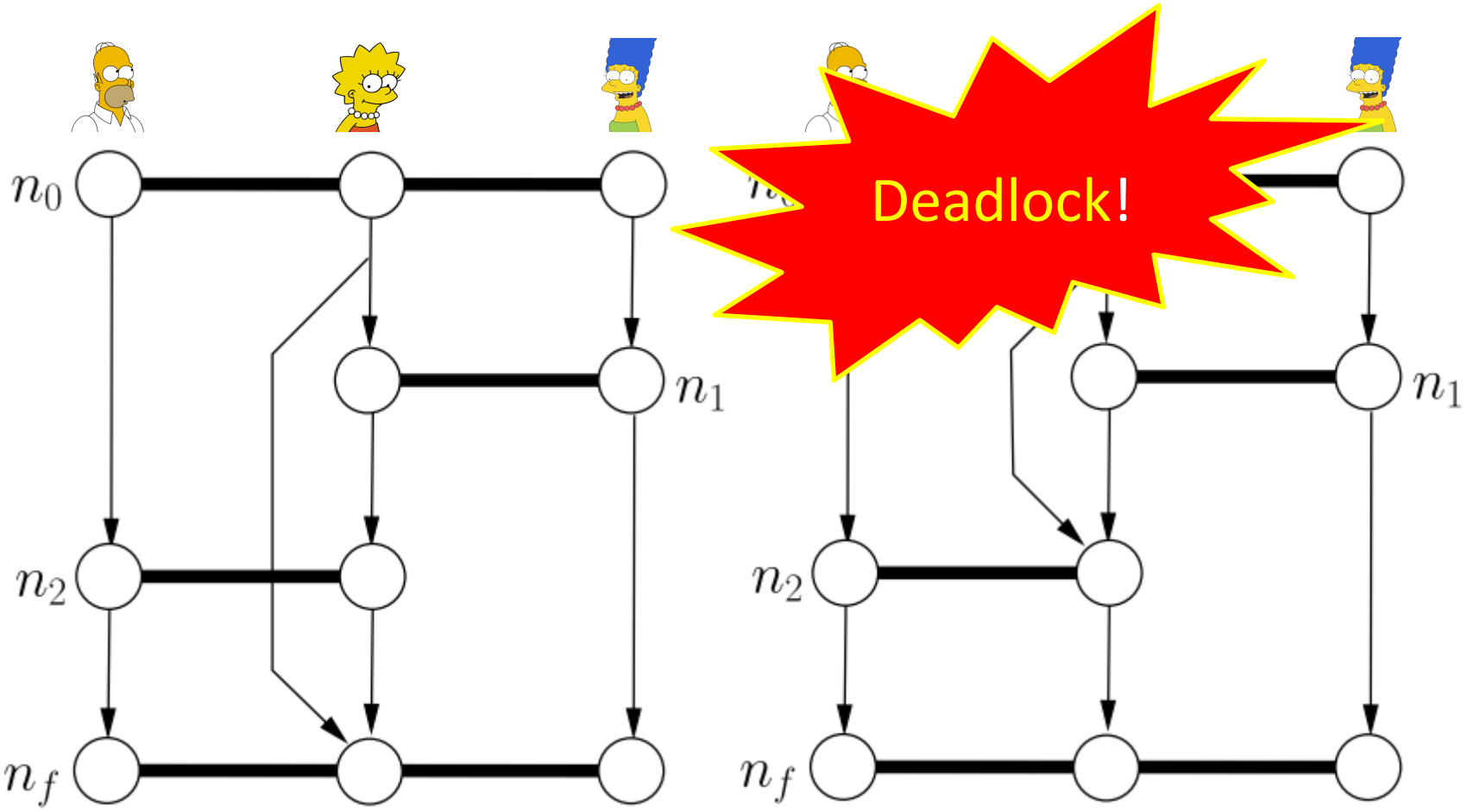


Negotiations → 1-safe Petri nets

- Places of the form
  [agent, set of atoms] ⇒
  number of places potentially
  exponential



Analysis problems: Soundness



Analysis problems: Soundness

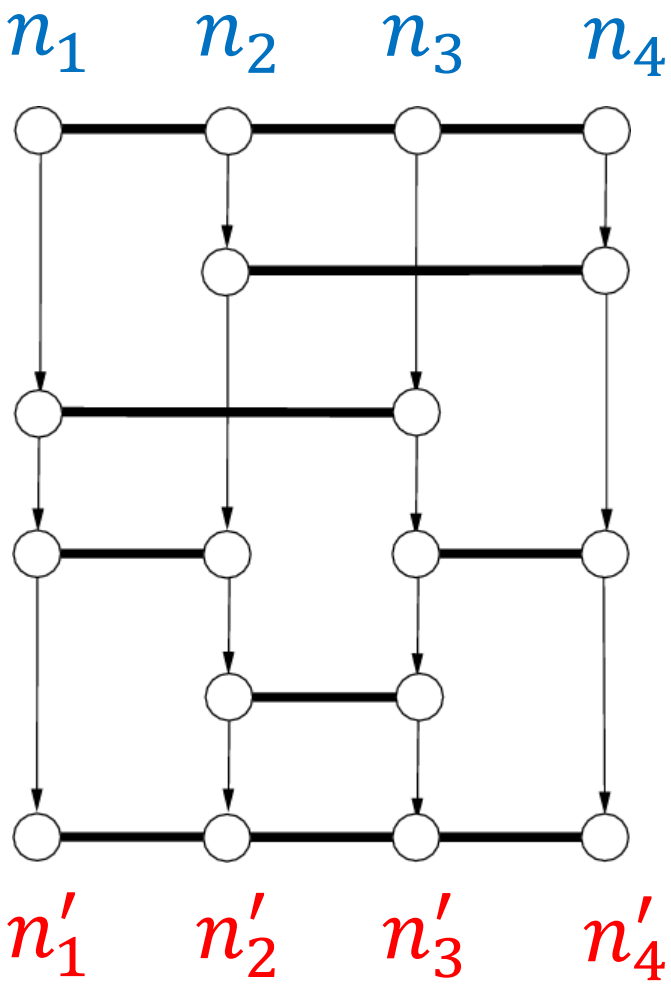
- Large step:
  sequence of occurrences of atoms starting with the initial atom and ending with the final atom.
- A negotiation is sound if
  1. Every execution can be extended to a large step.
  2. No useless atoms: every atom occurs in some large step.

  (cf. van der Aalst's workflow nets)
- In particular: soundness → deadlock-freedom

Analysis problems: Summarization

- Each large step induces a relation between initial and final global states
- The summary of a negotiation is the union of these relations (i.e., the whole input-output relation).
- The summarization problem consists of, given a sound negotiation, computing its summary.

Analysis problems: Summarization



Sorting negotiation correct if

- sound, and
- summary consists of all pairs

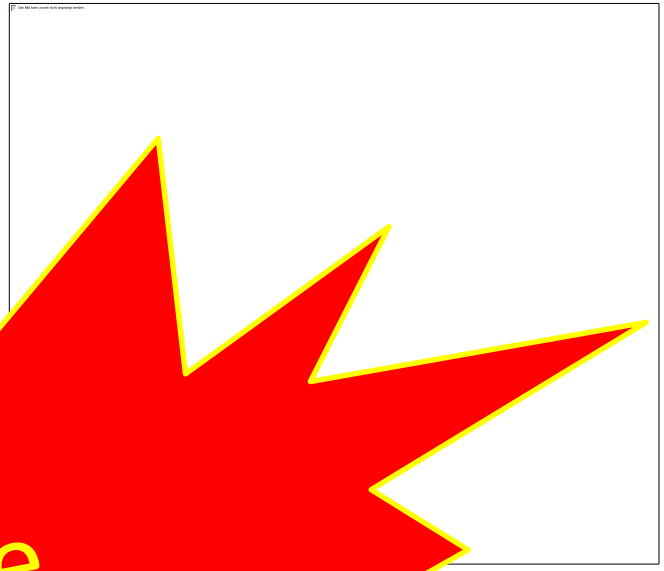
  ((n_1, n_2, n_3, n_4) , (n'_1, n'_2, n'_3, n'_4))

  such that $n'_1 n'_2 n'_3 n'_4$ is permutation of

  $n_1 n_2 n_3 n_4$ and $n'_1 \leq n'_2 \leq n'_3 \leq n'_4$

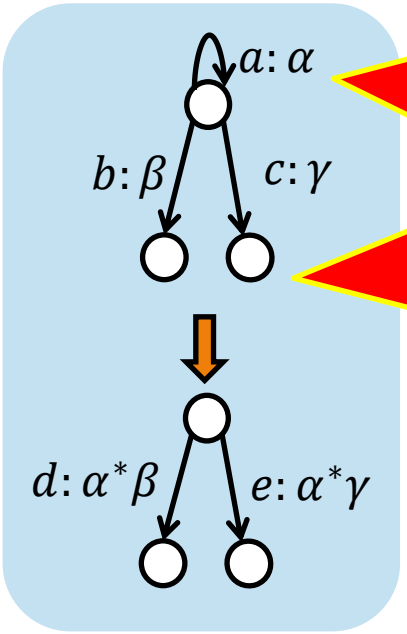
Computing summaries

- A simple algorithm to compute a summary:
  - Compute the LTS of the negotiation
  - Apply reduction rule

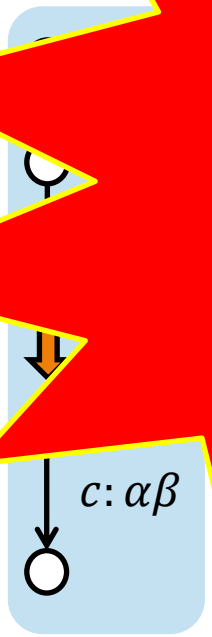


$a: \alpha$
$b: \beta$
...
$e: \epsilon$

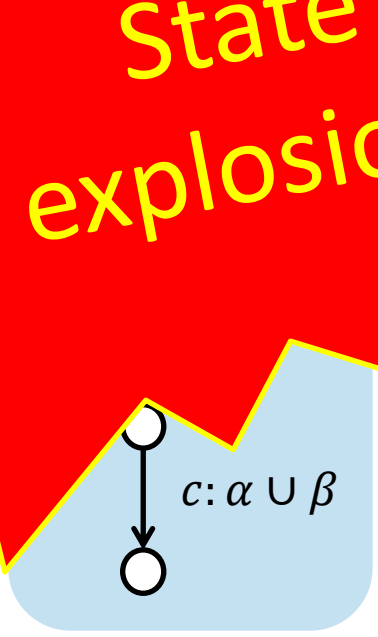
State explosion!



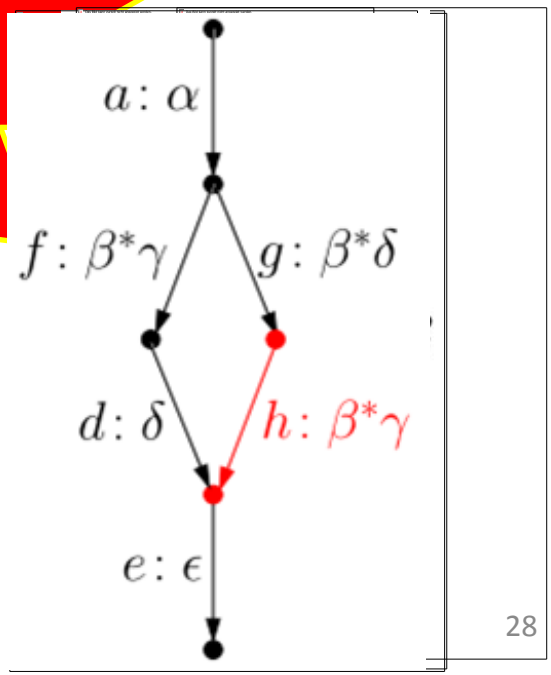
Iteration



Shortcut



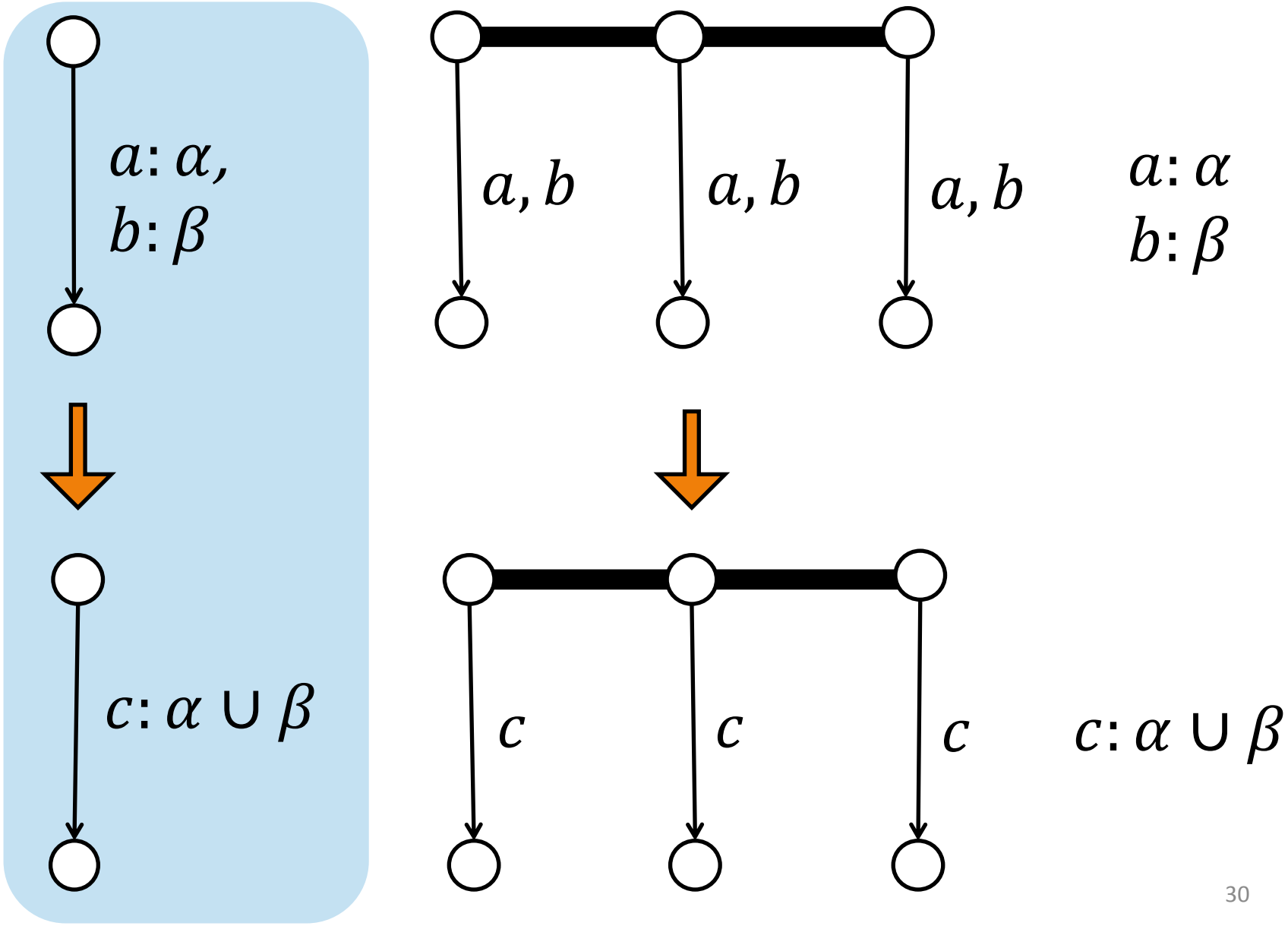
Merge



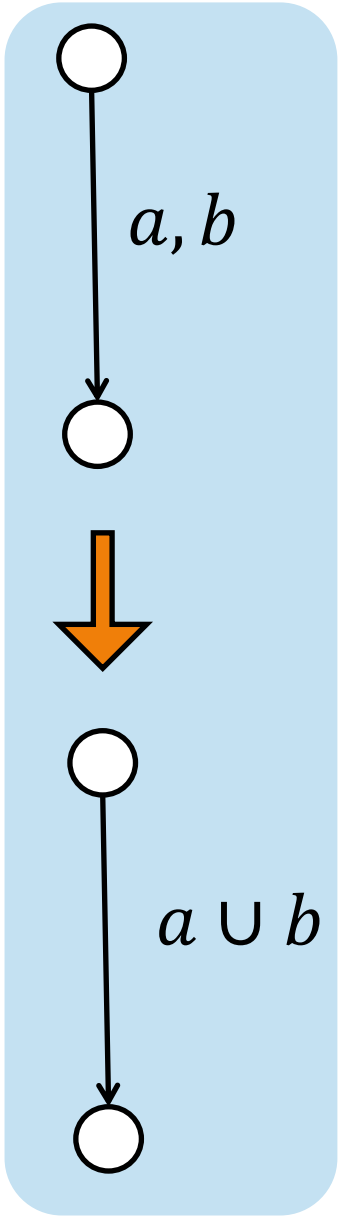
Reduction rules for negotiations

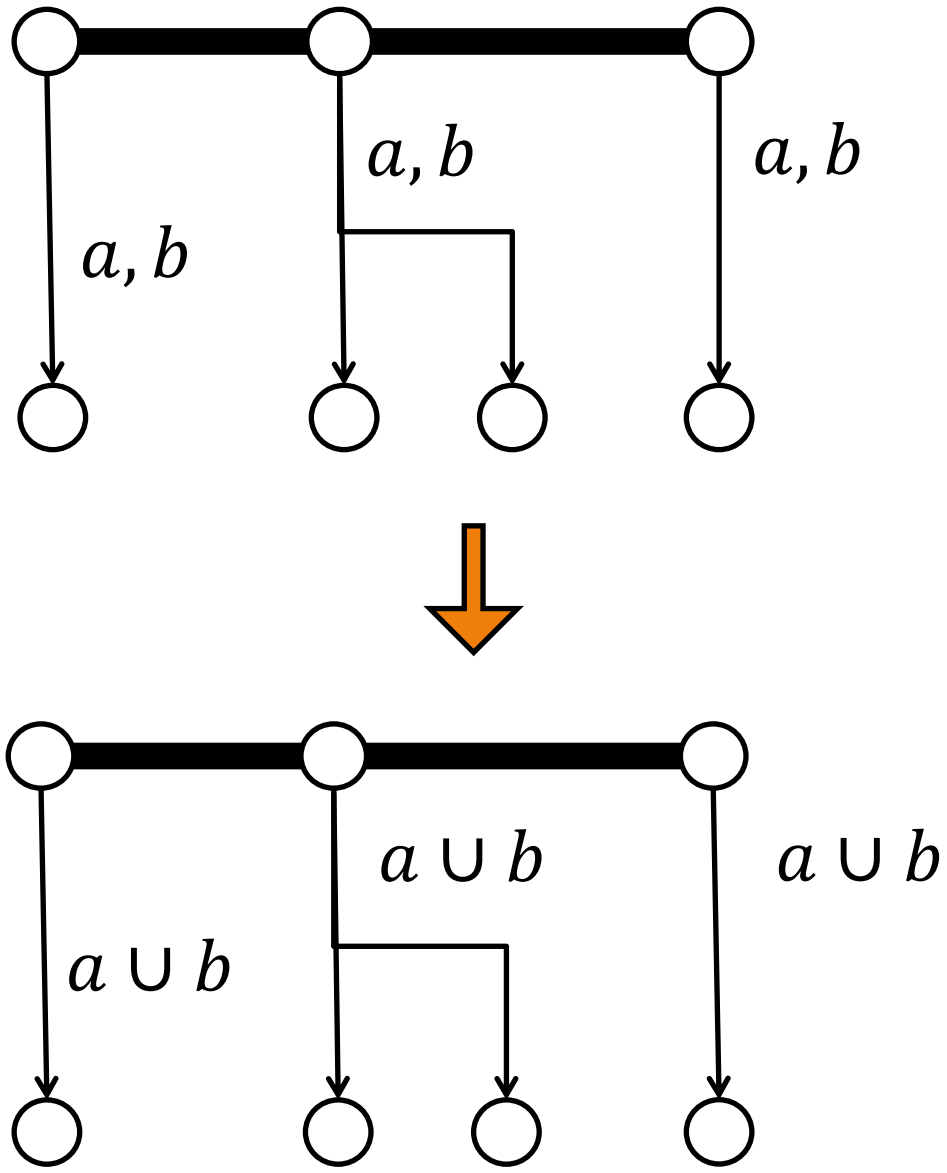
- Aim: find reduction rules
  acting directly on negotiation diagrams
  (instead of their transition systems).
- Rules must:
  - preserve soundness:
    sound after iff sound before
  - preserve the summary:
    summary after equal to summary before
- We look for local rules:
  application conditions and changes involve only a local neighbourhood of the application point.

Rule 1: Merge

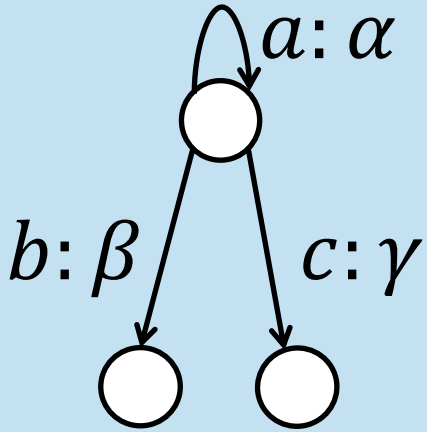

$a: \alpha,$
$b: \beta$
a, b
a, b
a, b
$a: \alpha$
$b: \beta$
$c: \alpha \cup \beta$
c
c
c
$c: \alpha \cup \beta$

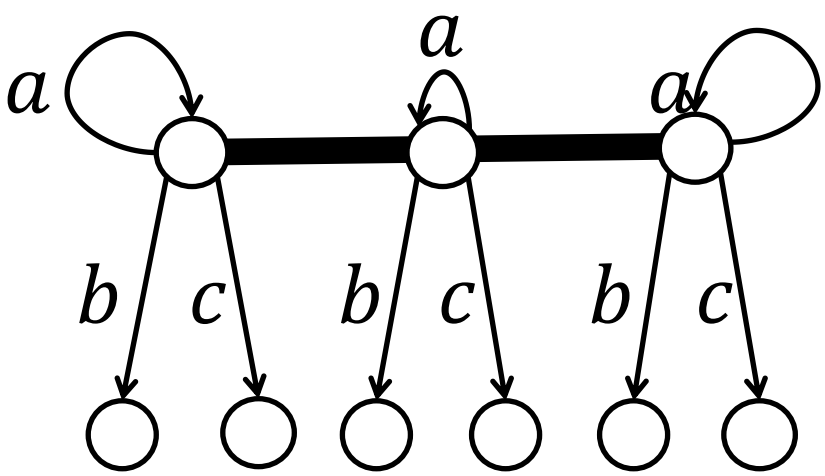
Rule 1: Merge



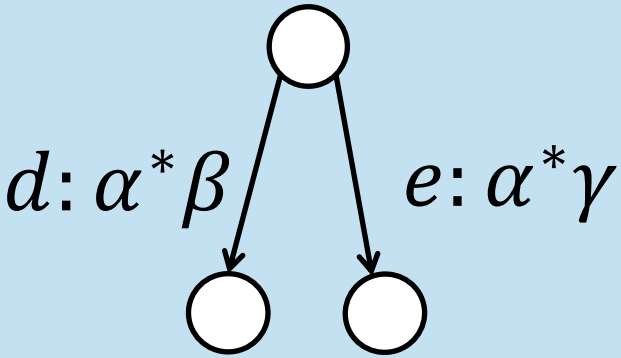


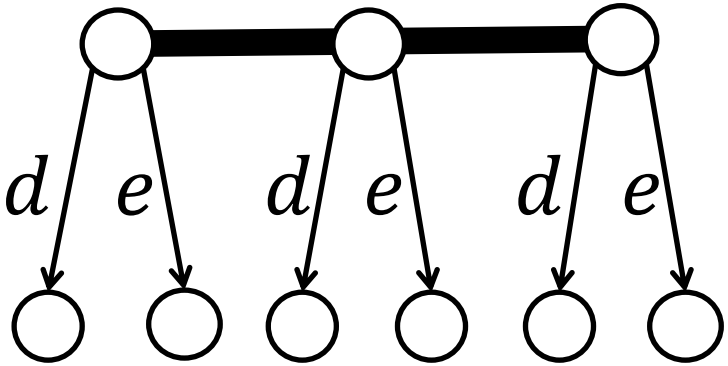
Rule 2: iteration





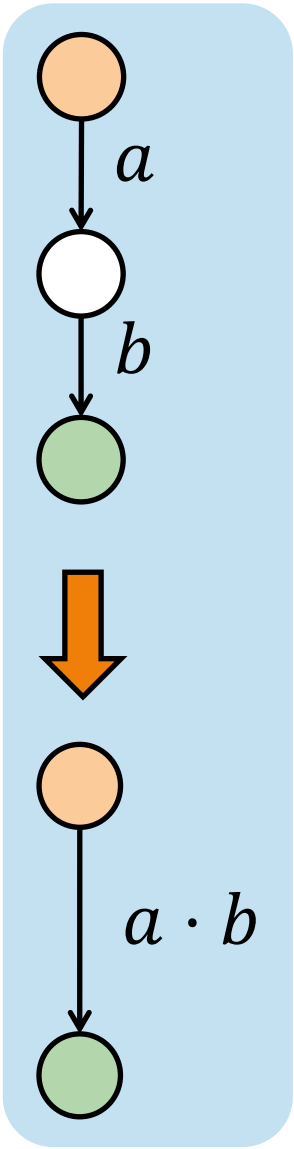
$a: \alpha$
$b: \beta$
$c: \gamma$

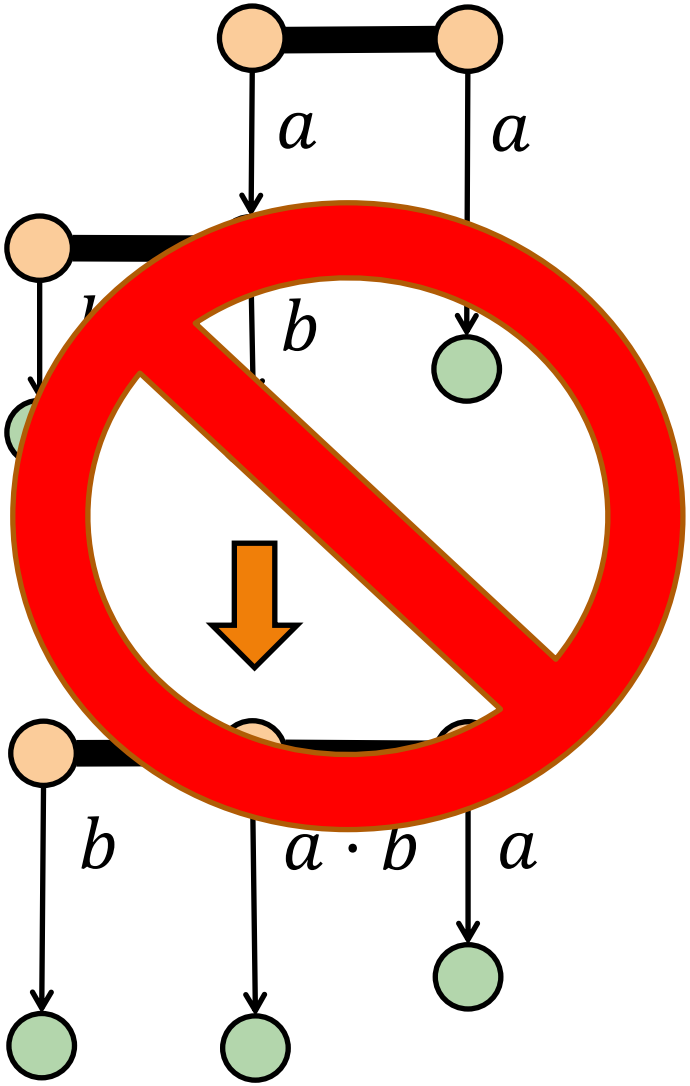


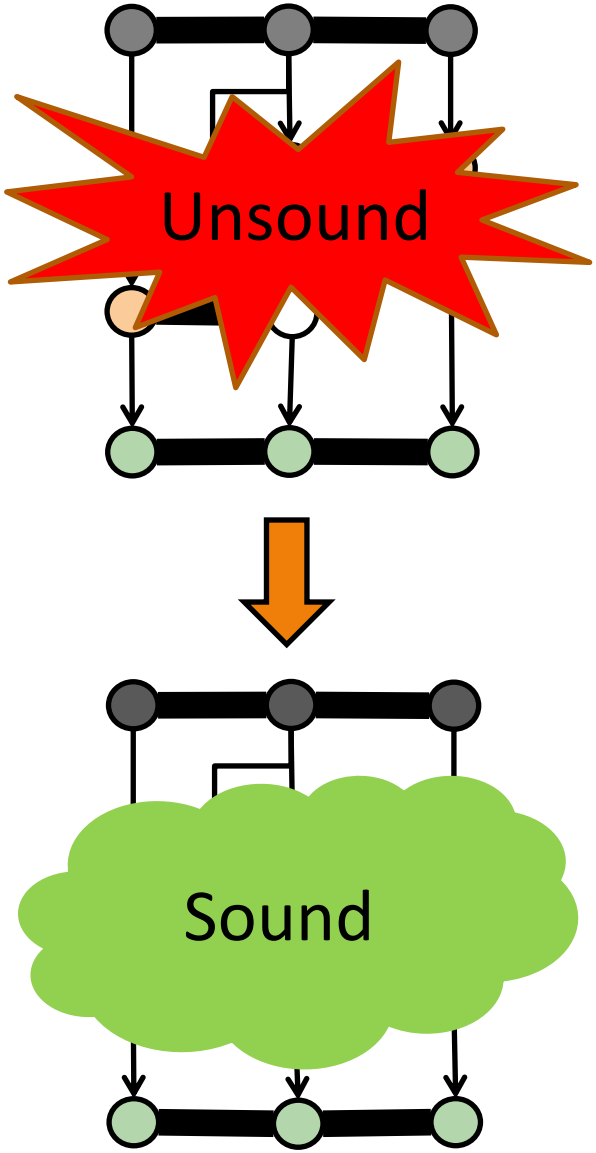


$d: \alpha^* \beta$
$e: \alpha^* \gamma$

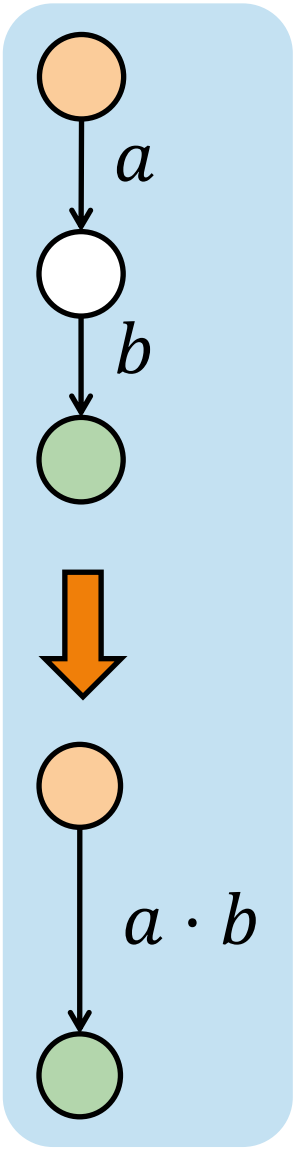
Rule 3: shortcut

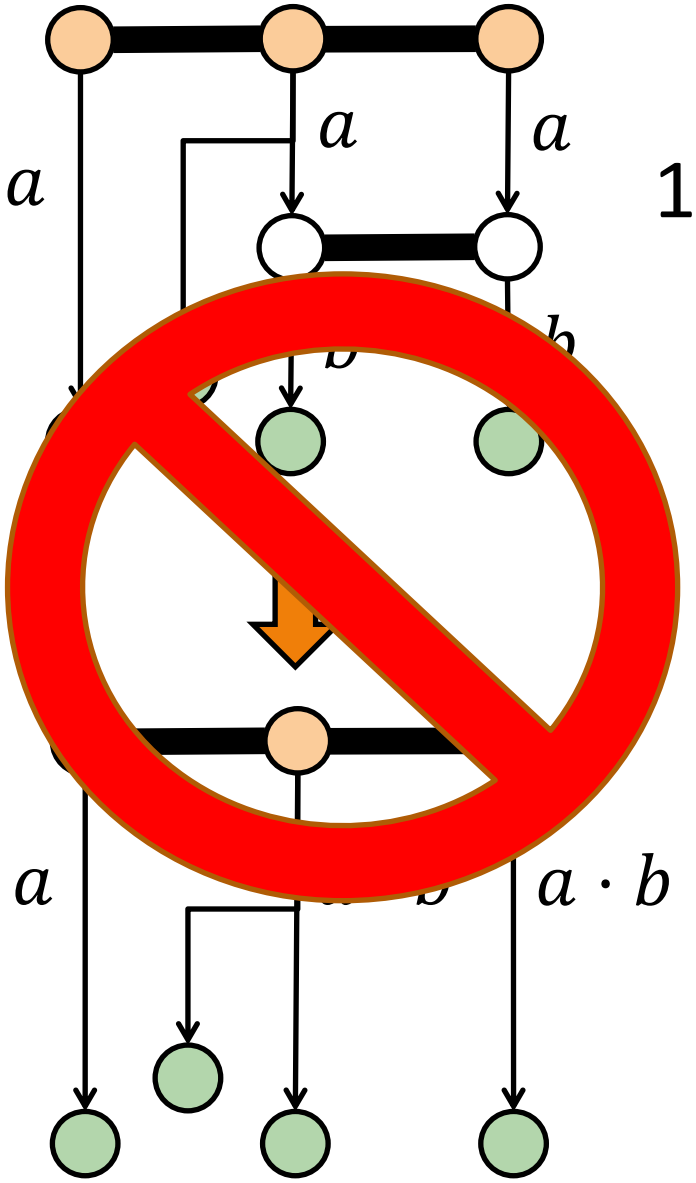




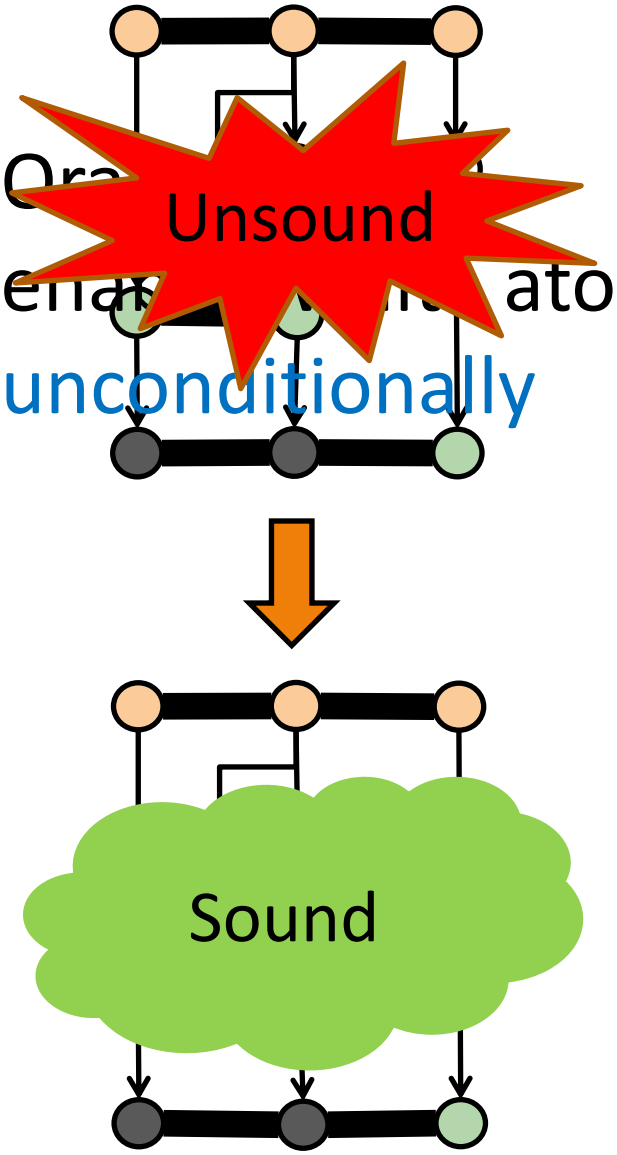


Rule 3: shortcut

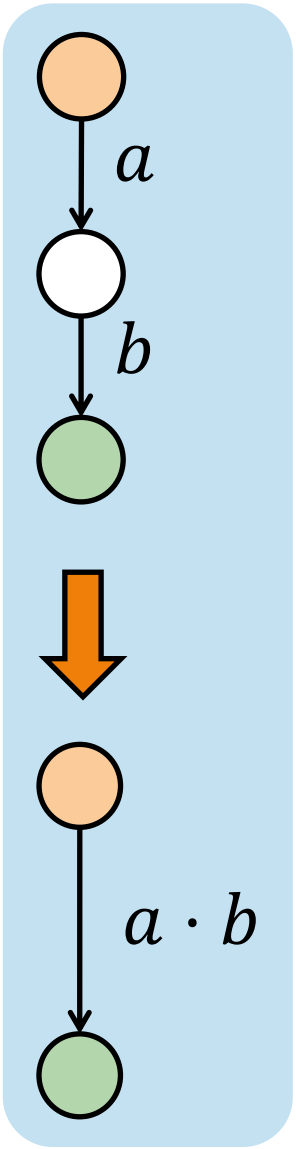




1. Ora[illegible] atom unconditionally

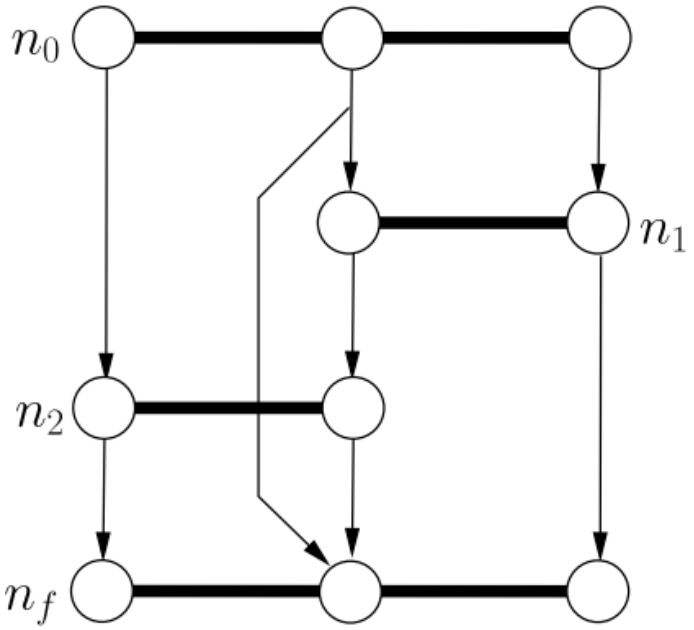


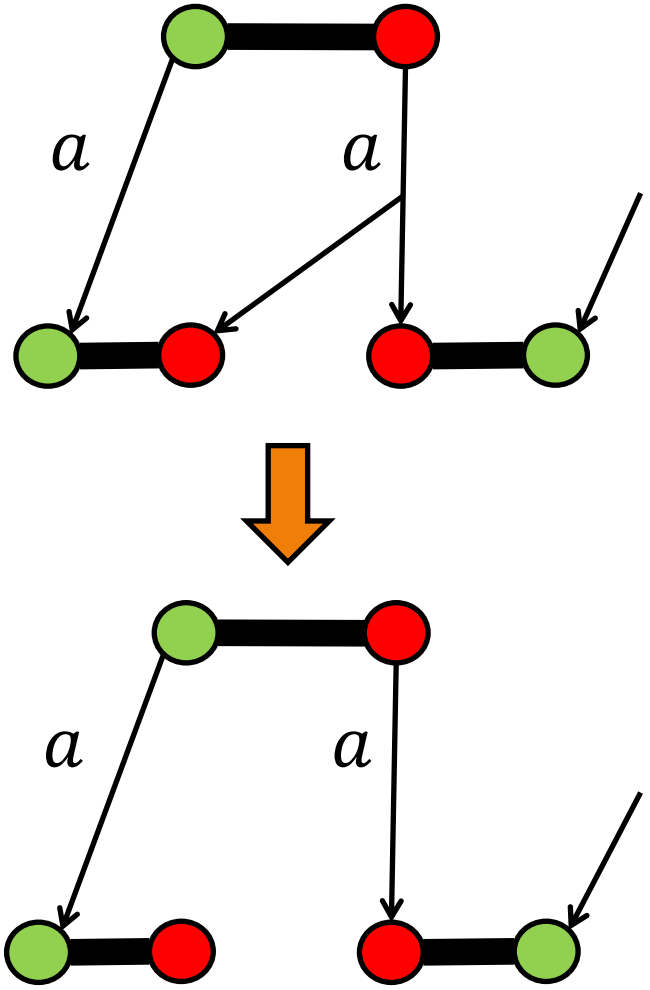
Rule 3: shortcut



range atom enables
hite atom
nconditionally.
o forks from orange
white.
ne more technical
ndition (see paper).

Rule 4: useless arc





Agent 1

Agent 2

Completeness and polynomiality

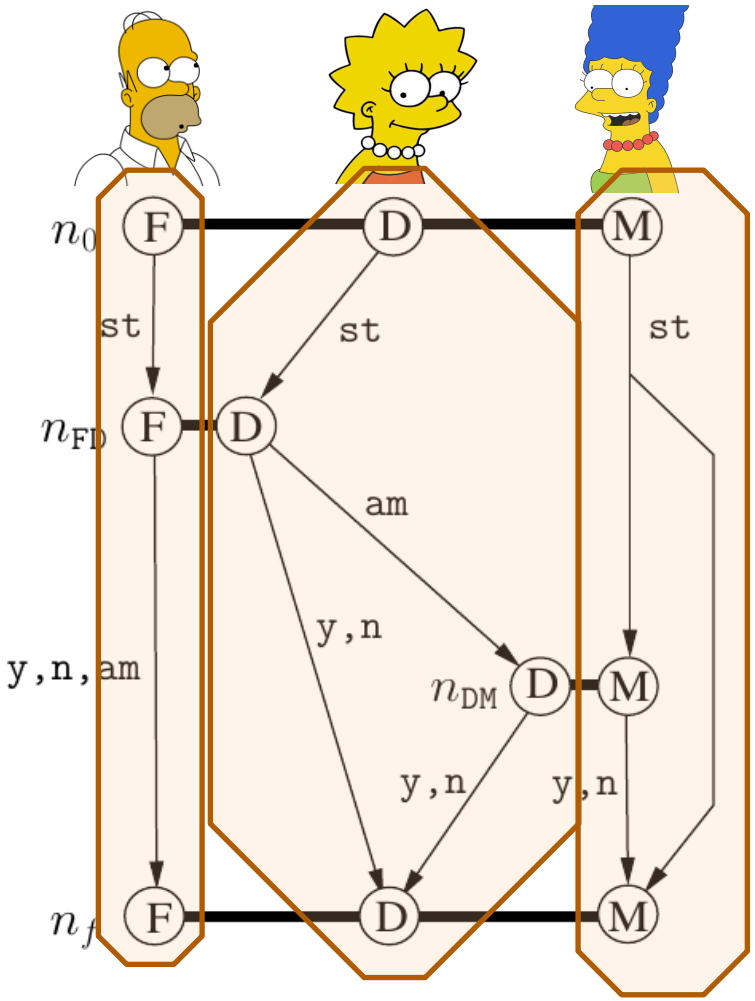
- A set of rules is complete for a class if it reduces all negotiations in the class to atomic negotiations.
- A complete set of rules is polynomial if every sound negotiation with k atoms is reduced to an atom by $p(k)$ rule applications, for some polynomial p.

Theorem [CONCUR 13]: deciding if a giving pair of global states belongs to the summary of a given negotiation is PSPACE-complete, even for every simple transformers.

Polynomiality and completeness results very unlikely for arbitrary negotiations.

Deterministic agents and negotiations

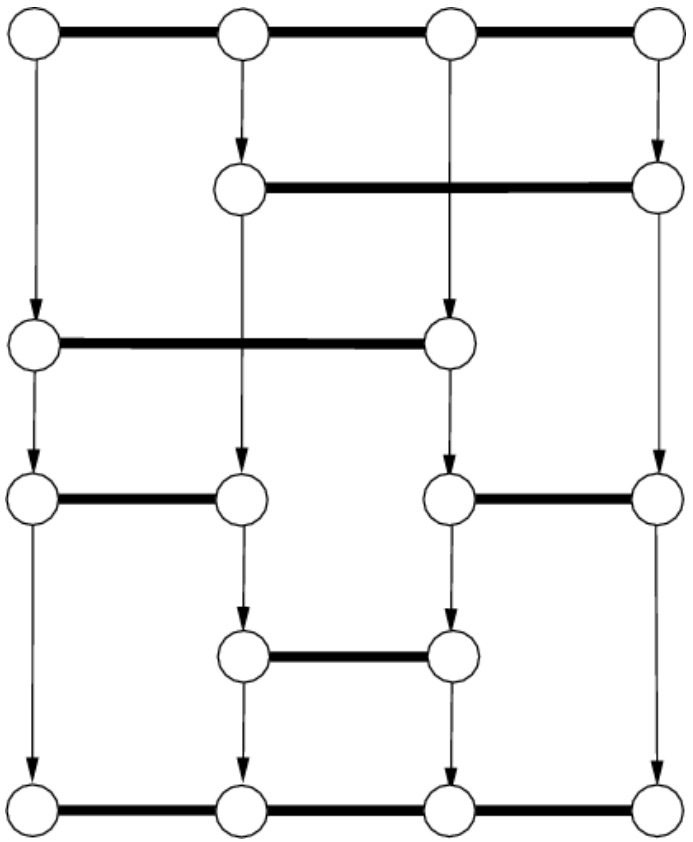
- An agent is deterministic if it is never ready to engage in more than one atom („no forks“).
- A negotiation is deterministic if every agent is deterministic.



non-deterministic

Deterministic agents and negotiations

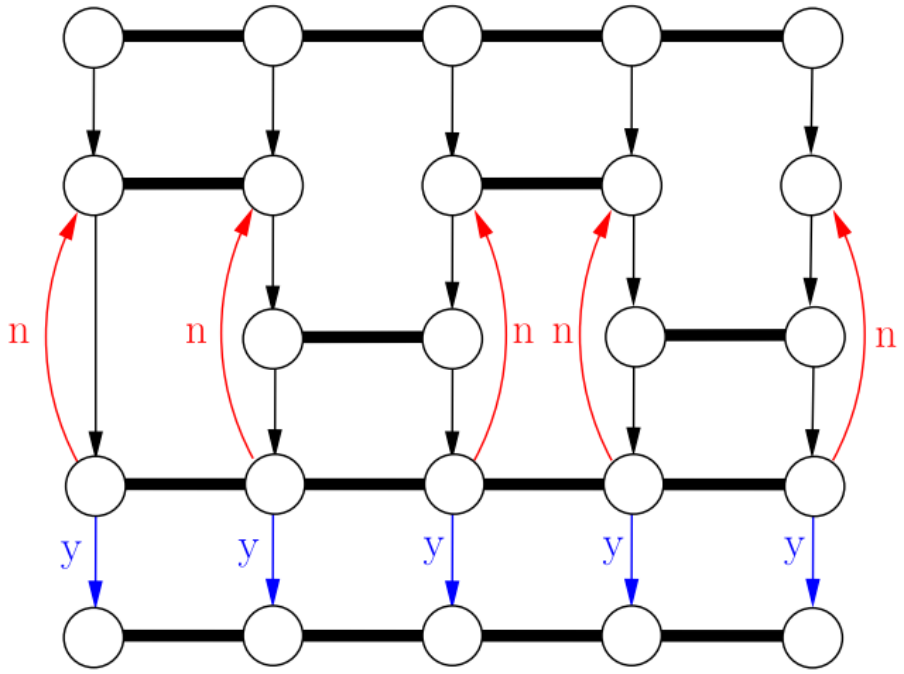
- An agent is deterministic if it is never ready to engage in more than one atom („no forks“).
- A negotiation is deterministic if every agent is deterministic.



deterministic

Deterministic agents and negotiations

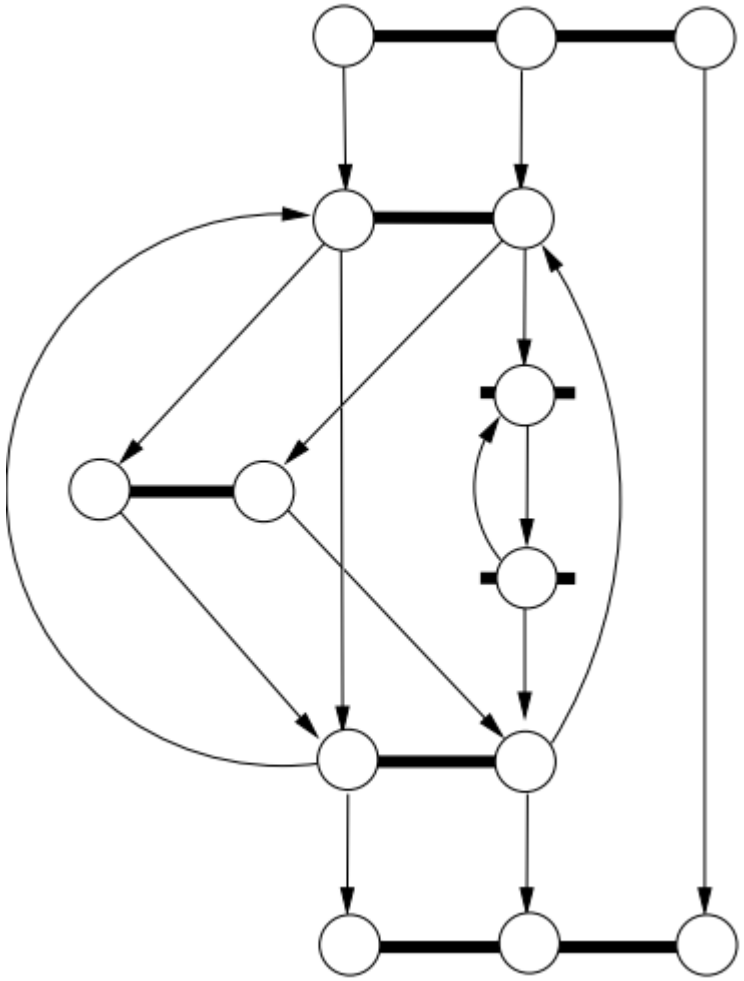
- An agent is deterministic if it is never ready to engage in more than one atom („no forks“).
- A negotiation is deterministic if every agent is deterministic.



deterministic

Deterministic agents and negotiations

- An agent is deterministic if it is never ready to engage in more than one atom („no forks“).
- A negotiation is deterministic if every agent is deterministic



deterministic

Deterministic negotiations: Results

- Theorem [CONCUR 13]:

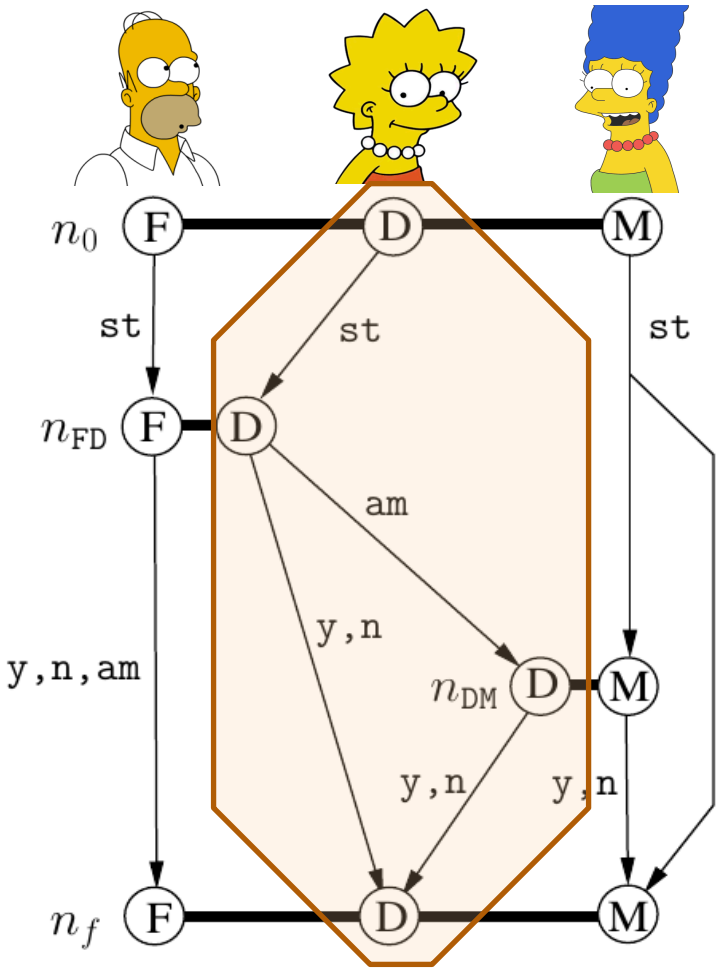
  The shortcut and merge rules are complete and polynomial for acyclic deterministic negotiations

- Theorem [FOSSACS 14]:

  The shortcut, merge, and iteration rules are complete and polynomial for arbitrary deterministic negotiations

Weakly deterministic negotiations

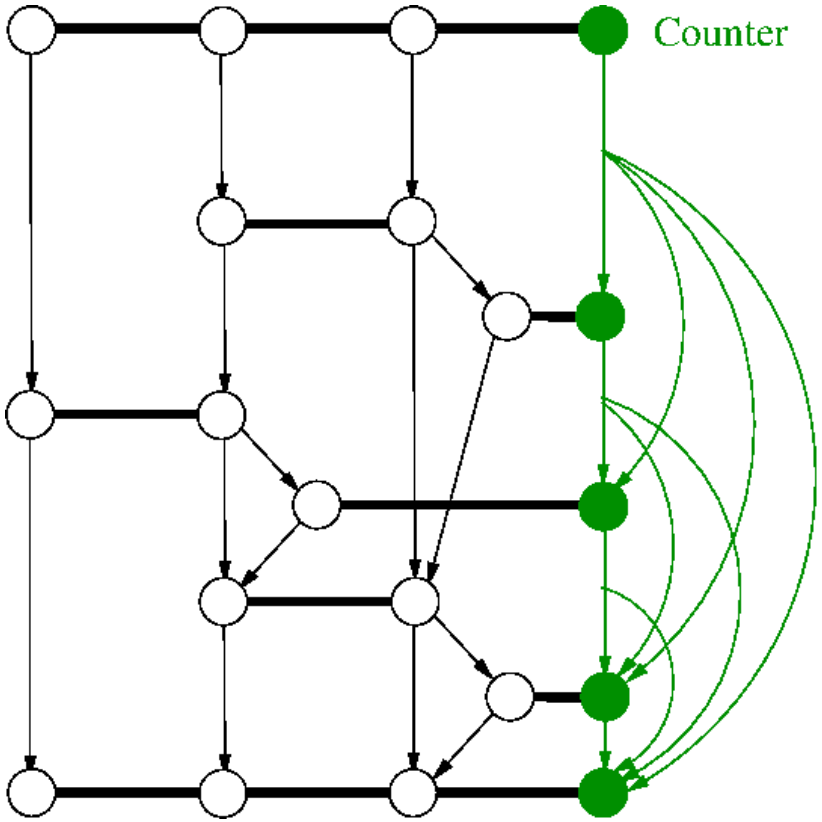
- A negotiation is weakly deterministic if every atom has a deterministic party



weakly deterministic

Weakly deterministic negotiations

- A negotiation is weakly deterministic if every atom has a deterministic party



weakly deterministic

Weakly Deterministic Negotiations: Results

- Theorem [CONCUR 13]:

  The shortcut, merge, and useless arc rules are complete for acyclic, weakly deterministic negotiations.

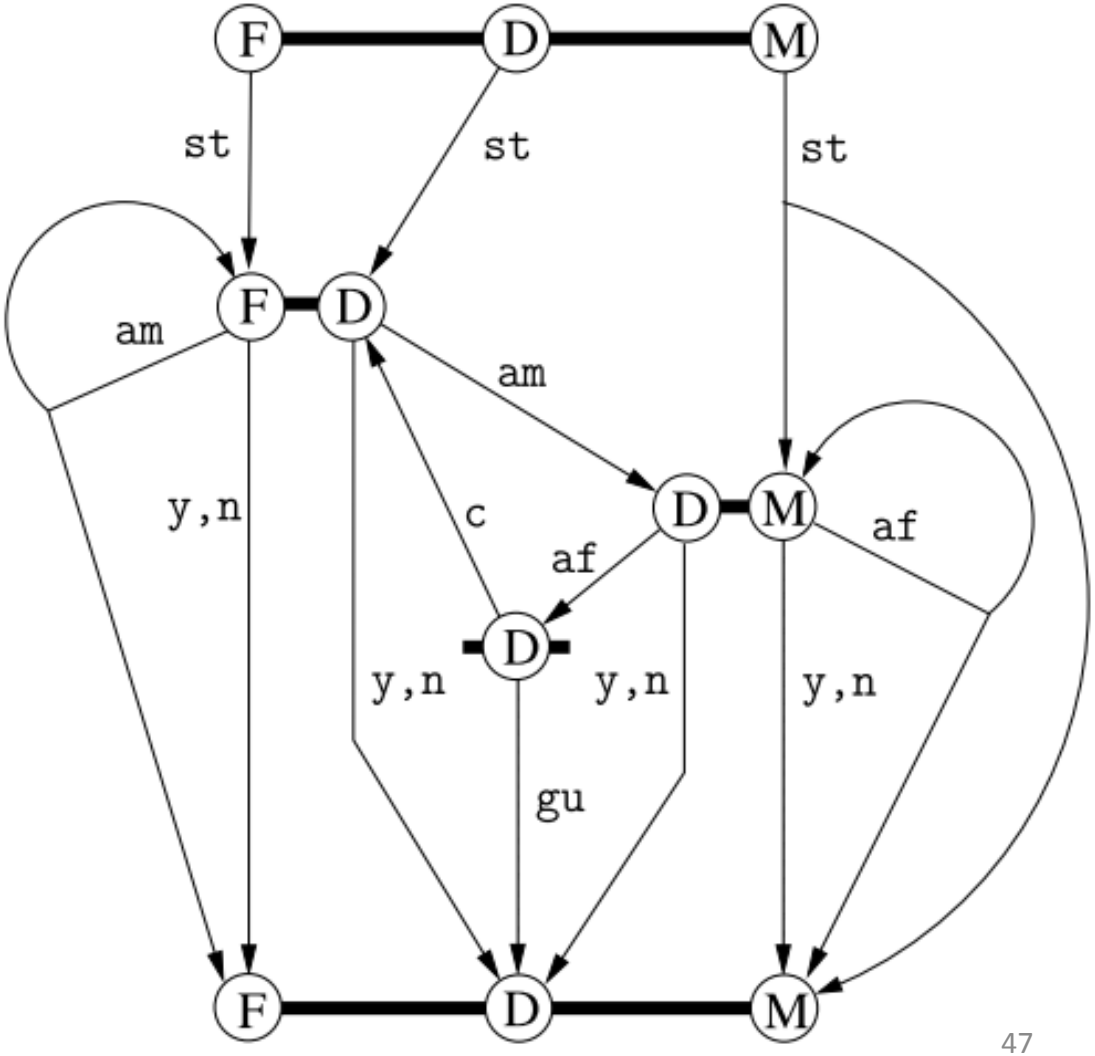
Deterministic negotiations: Results

- Theorem [FOSSACS 14]:

  The shortcut, merge, and iteration rules are complete and polynomial for arbitrary deterministic negotiations

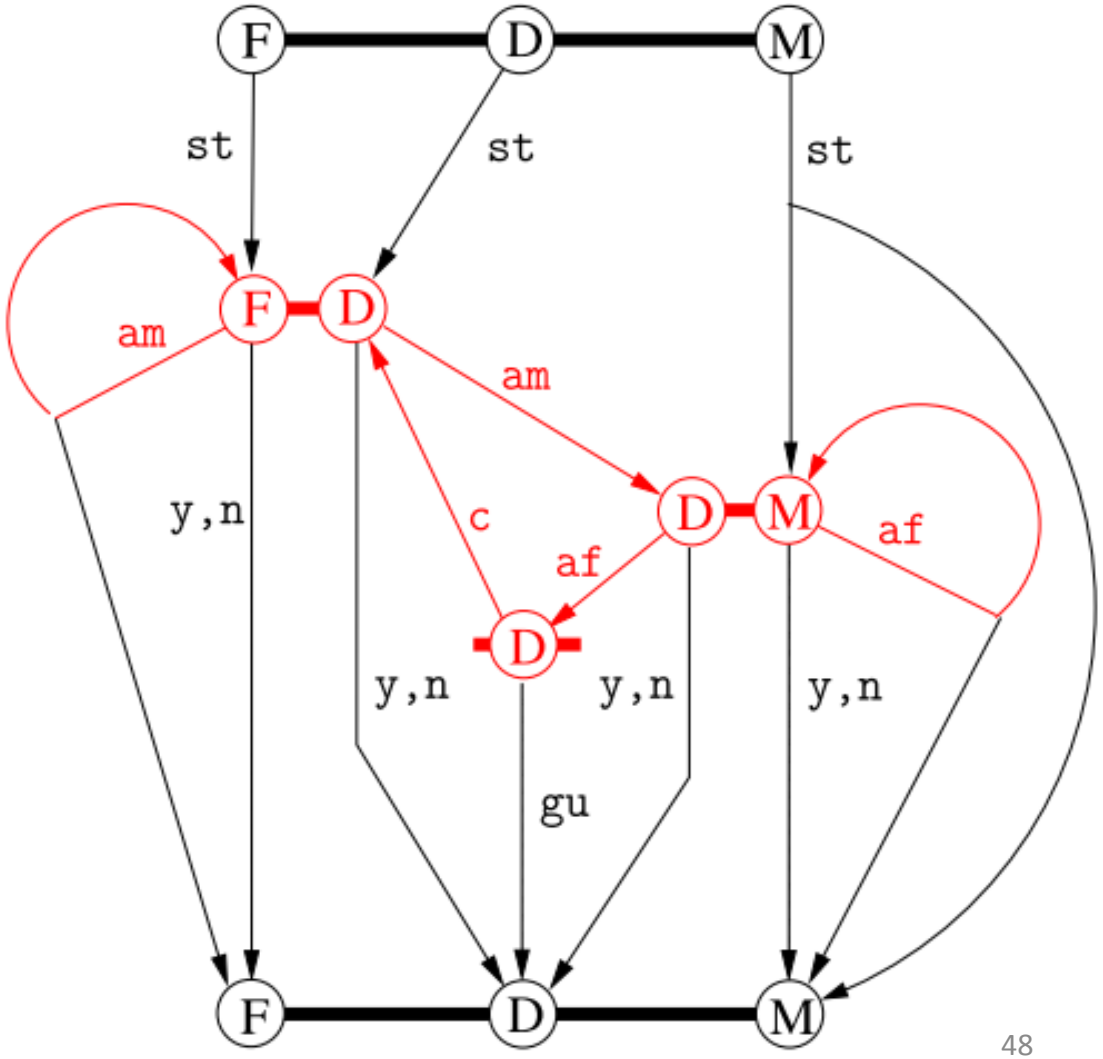
A Proof Sketch: Loop fragments

- A loop is a minimal firing sequence leading from a reachable marking to itself.



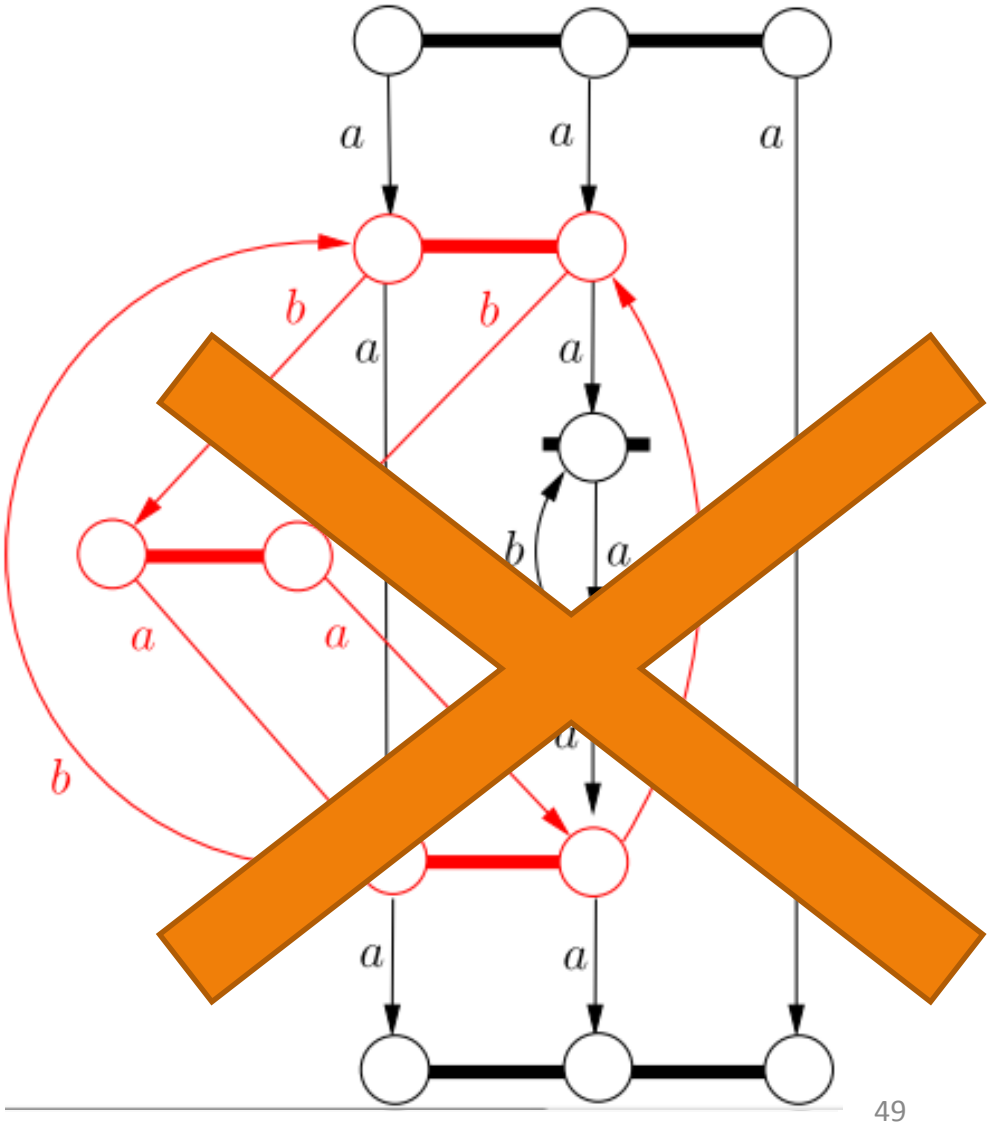
A Proof Sketch: Loop fragments

- A loop is a minimal firing sequence leading from a reachable marking to itself.
- A loop fragment is the subnegotiation that „occurs in the loop“



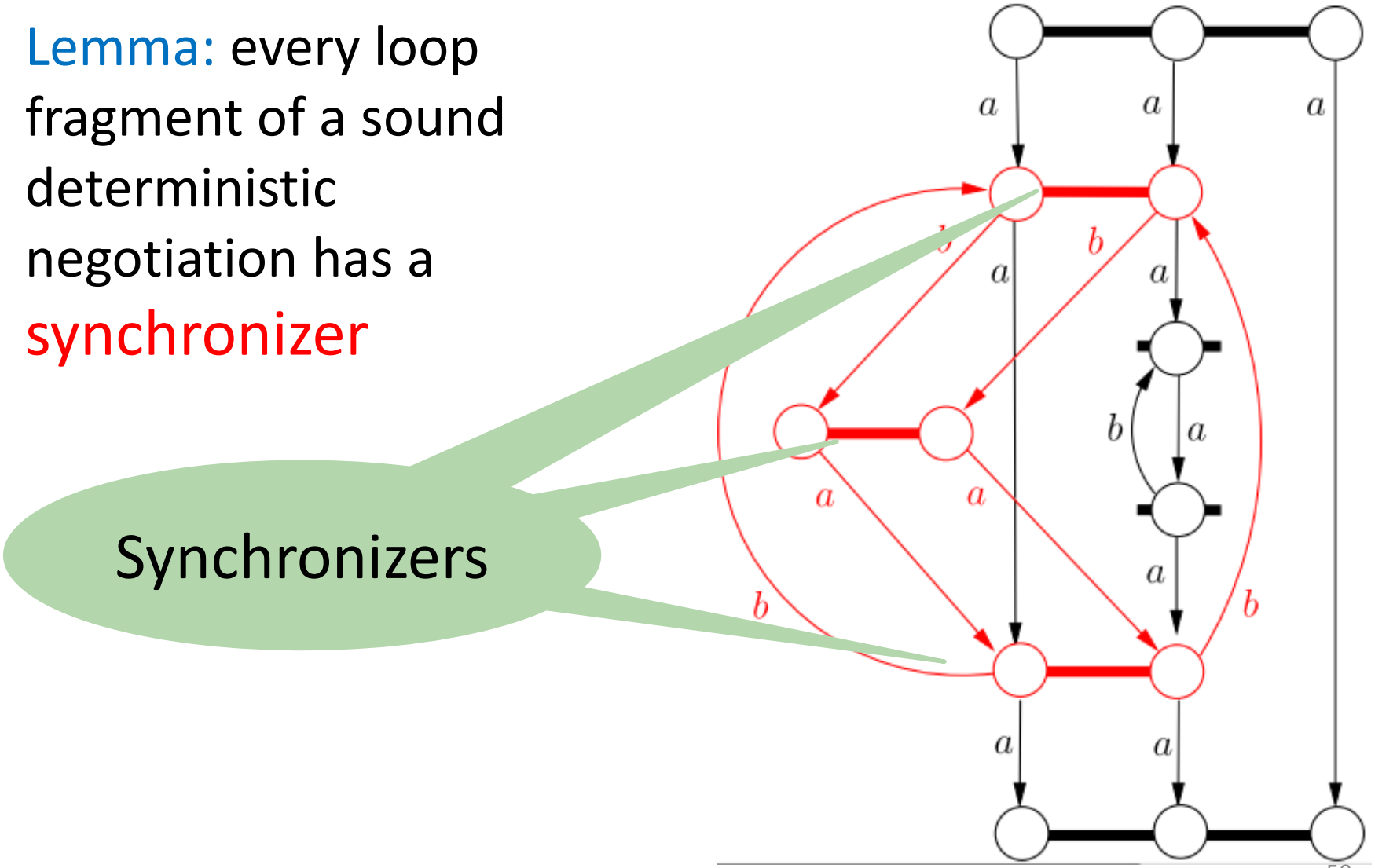
A Proof Sketch: Loop fragments

- A loop is a minimal firing sequence leading from a reachable marking to itself.
- A loop fragment is the subnegotiation that „occurs in the loop“



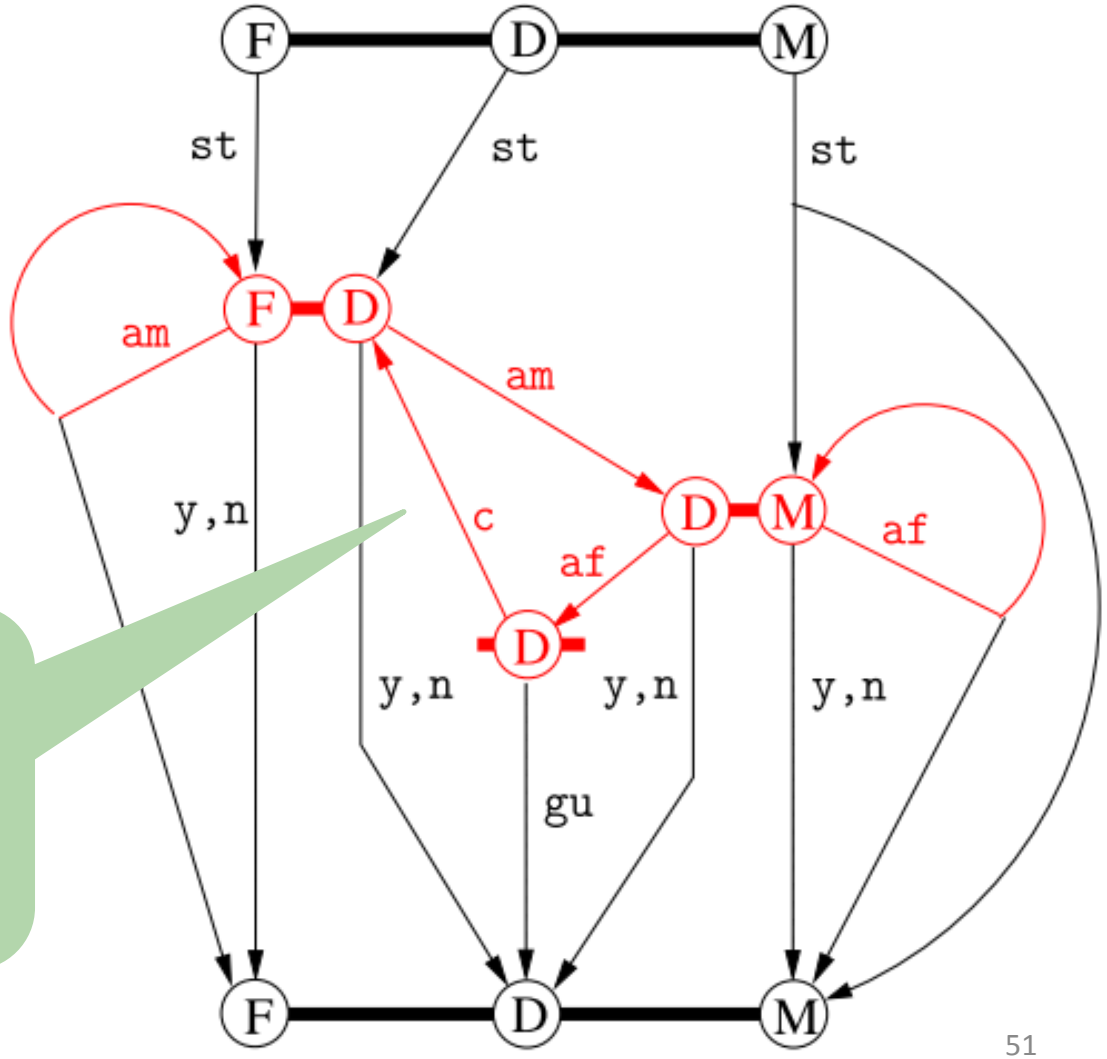
A Proof Sketch: Loop fragments

- Lemma: every loop fragment of a sound deterministic negotiation has a synchronizer



A Proof Sketch: Loop fragments

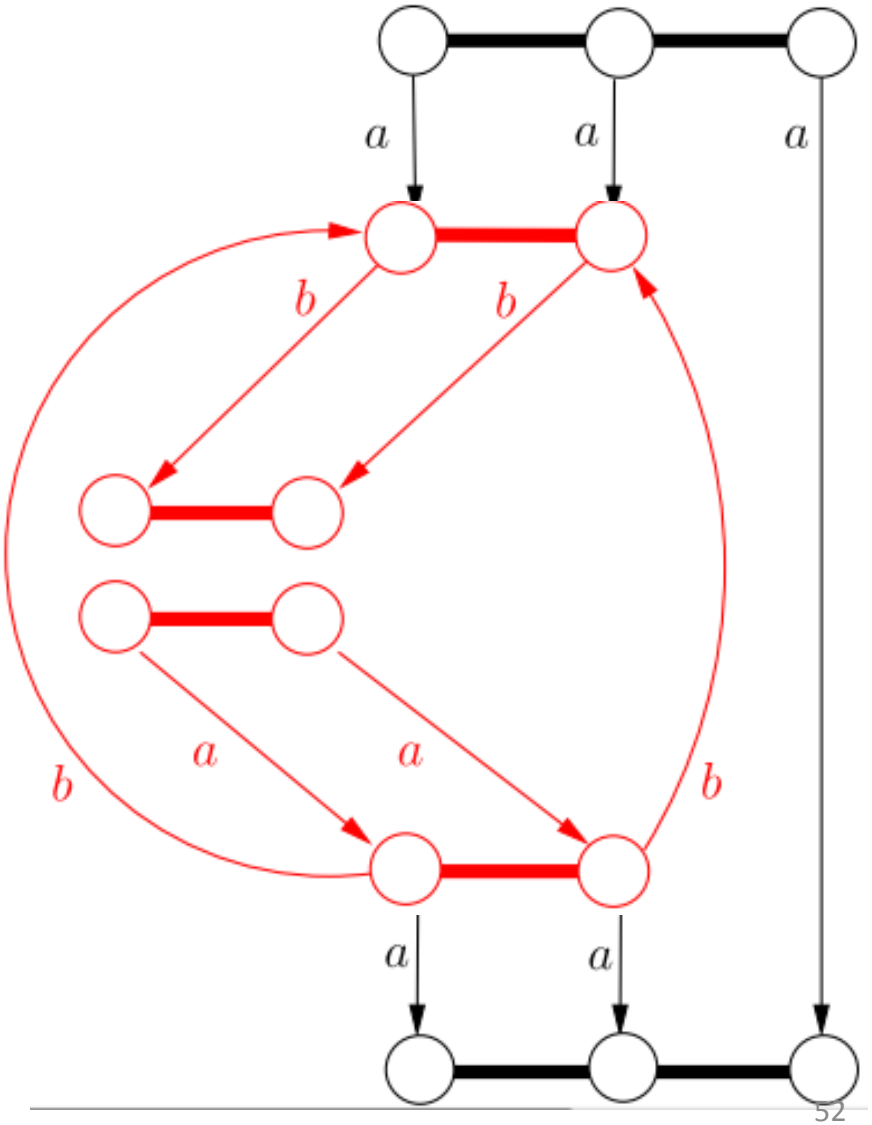
- Lemma: every loop fragment of a sound deterministic negotiation has a synchronizer



loop has no synchronizer, but negotiation is not deterministic

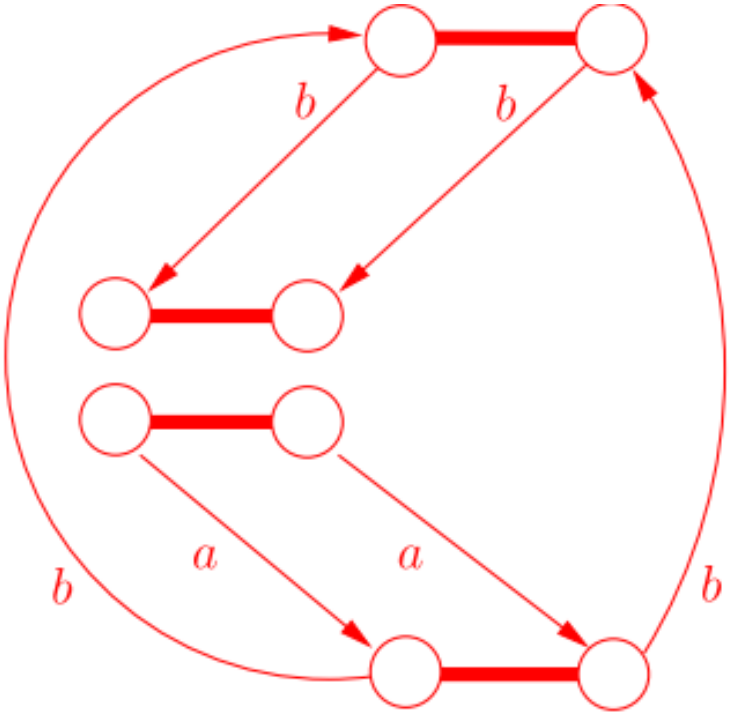
A Proof Sketch: Loop fragments

- Almost acyclic loop fragment: loop fragment that becomes acyclic after „cutting it along a synchronizer“



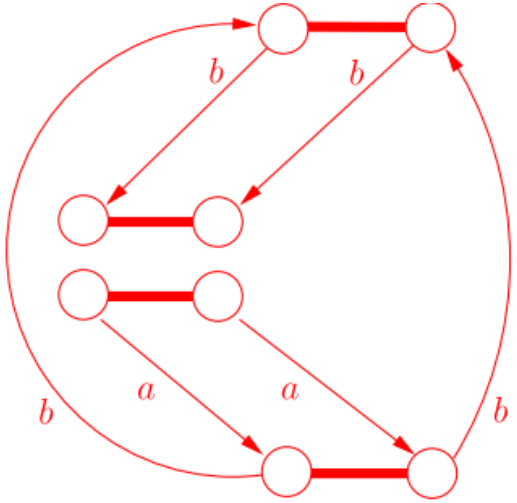
A Proof Sketch: Loop fragments

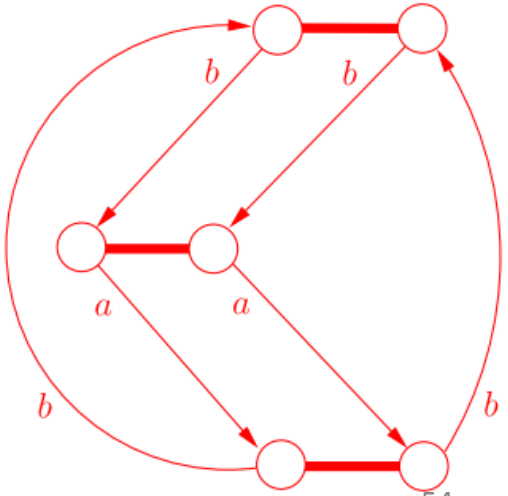
- Almost acyclic loop fragment: loop fragment that becomes acyclic after „cutting it along a synchronizer“
- Lemma: every cyclic, sound and deterministic negotiation contains an almost acyclic loop fragment.



A Proof Sketch: Loop fragments

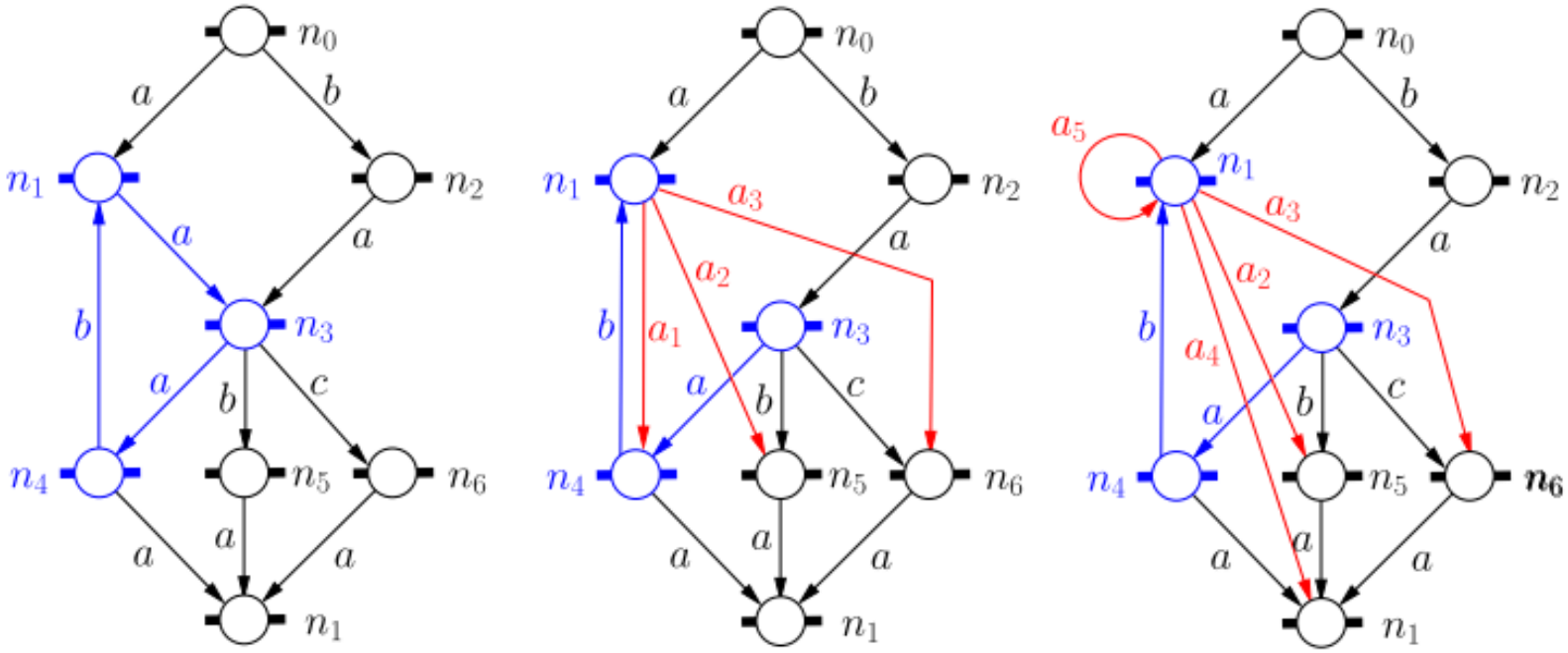
- Theorem [CONCUR 13]: Acyclic sound and deterministic negotiations can be reduced using the shortcut and merge rules.
- Corollary: loop fragments can be reduced to a „self-loop“ using the shortcut and merge rules. The self-loop can be reduced with the iteration rule.





A Proof Sketch: Polynomiality

- Problem: the reduction of a loop fragment to a self-loop produces „side-effects“



- Careful analysis required to ensure polynomialty.

A Proof Sketch: Polynomiality

- Theorem: a sound and deterministic negotiation with n atoms and k outcomes can be reduced by means of $O(n^4 k)$ applications of the shortcut, merge, and iteration rules.

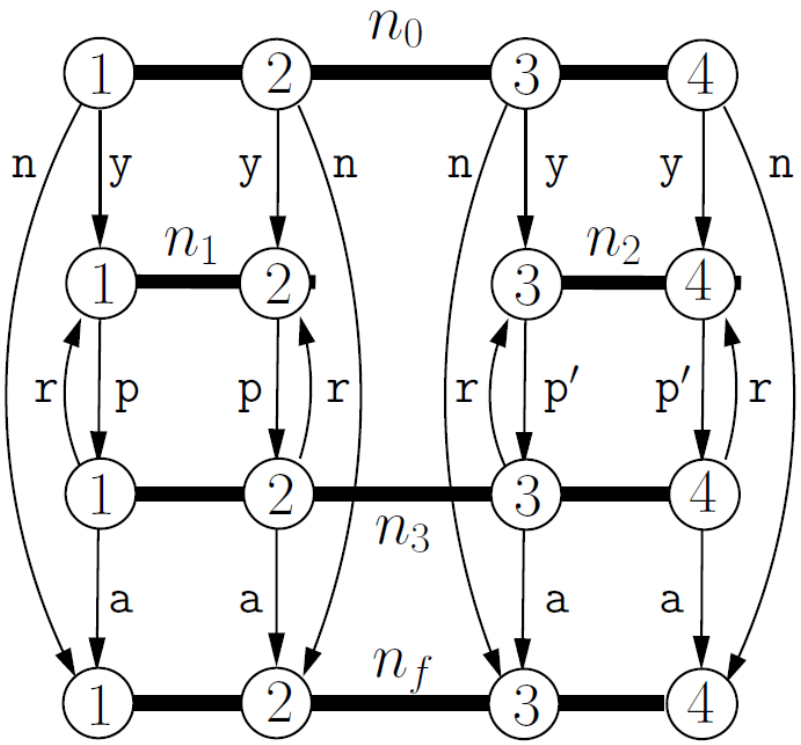
- Conjecture: $O(n^2 k)$ applications suffice.

New paper: Negotiation Programs

How can we implement negotiations?

How can we implement sound negotiations? (correct by construction)

Negotiation Programs: an example



n_0: discuss a proposal?

n_1 : team A makes plan.

n_2 : team B makes plan.

n_3 : plans are consistent?

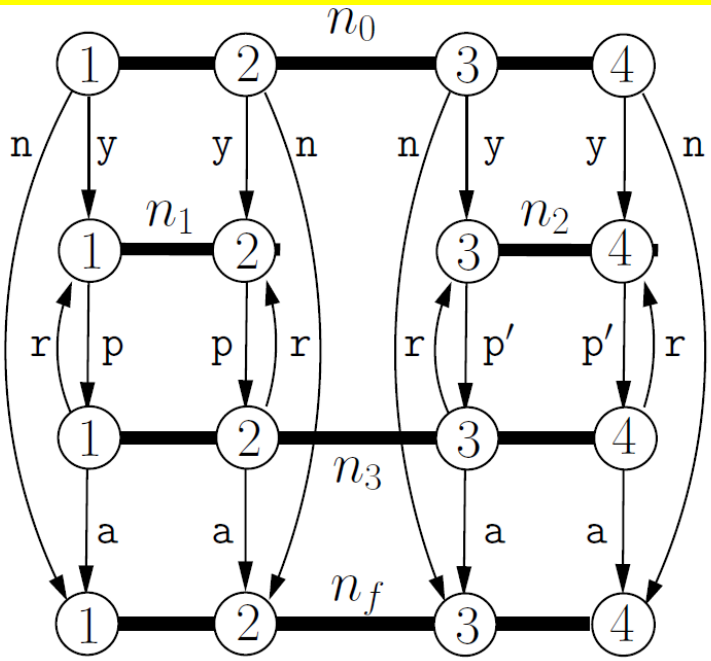
n_f : termination

Negotiation Programs: an example

agent a_1, a_2, a_3, a_4
actions $\mathtt{y}, \mathtt{n}, \mathtt{a}, \mathtt{r} : \{a_1, \ldots, a_4\}; \mathtt{p} : \{a_1, a_2\}; \mathtt{p}' : \{a_3, a_4\}$

do [] y : $(\mathtt{p} \parallel \mathtt{p}') \circ$
do a : **end** [] r : $(\mathtt{p} \parallel \mathtt{p}')$ **loop od** **end**
[] n : **end**
od

Negotiation Programs: an example

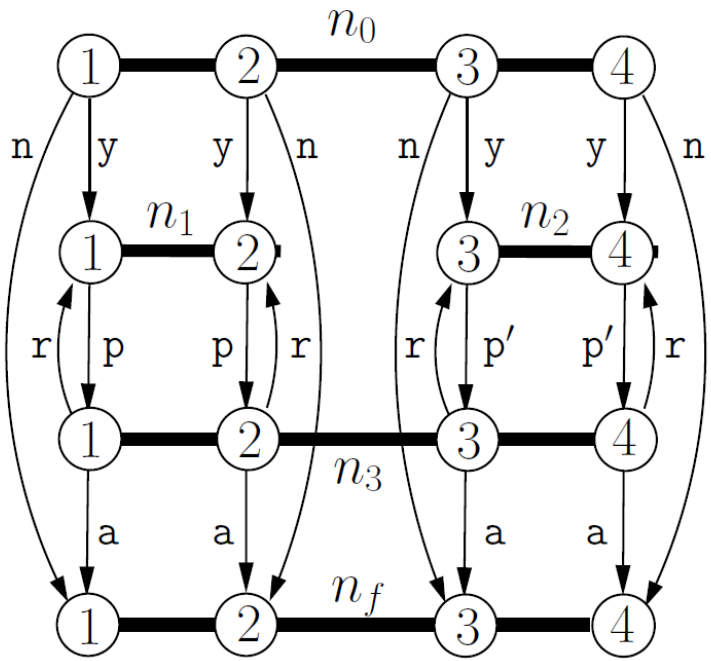


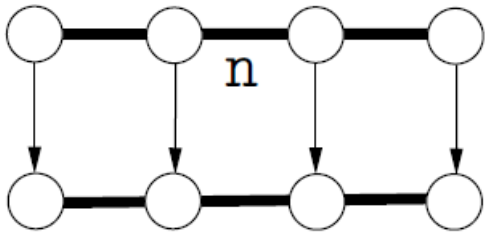
agent a_1, a_2, a_3, a_4

actions $\mathtt{y}, \mathtt{n}, \mathtt{a}, \mathtt{r} : \{a_1, \ldots, a_4\}; \mathtt{p} : \{a_1, a_2\}; \mathtt{p}' : \{a_3, a_4\}$

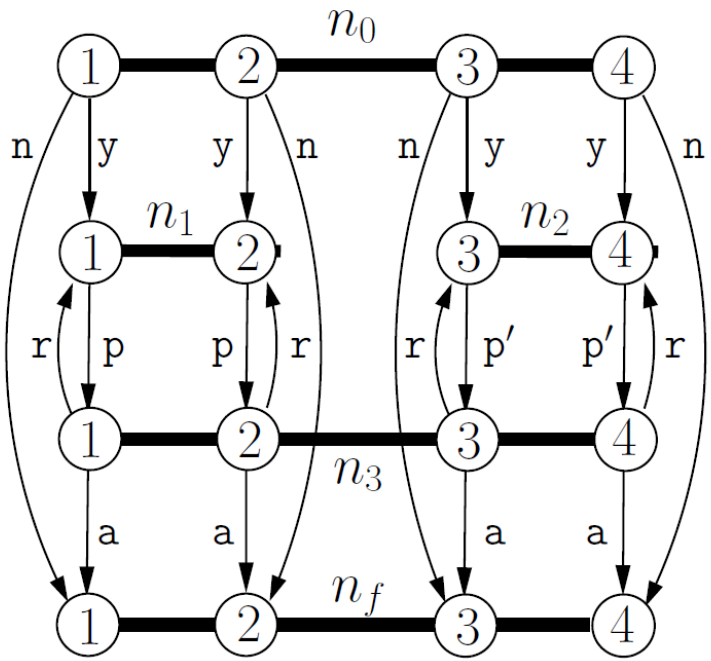
do [] y : $(\mathtt{p} \parallel \mathtt{p}')$ ∘
do a: **end** [] r: $(\mathtt{p} \parallel \mathtt{p}')$ **loop od** **end**
[] n : **end**
od

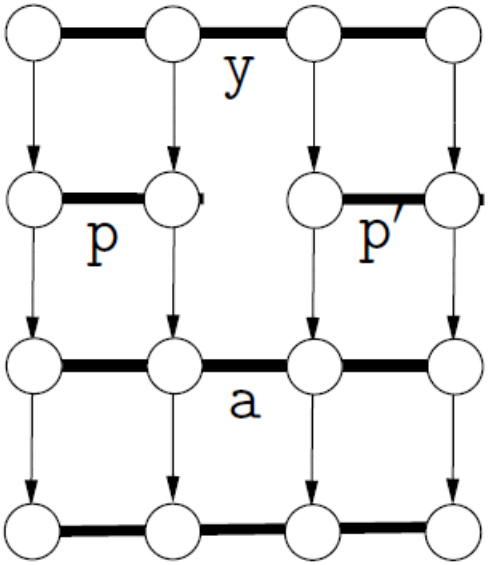
Semantics of negotiations: Mazurkiewicz traces



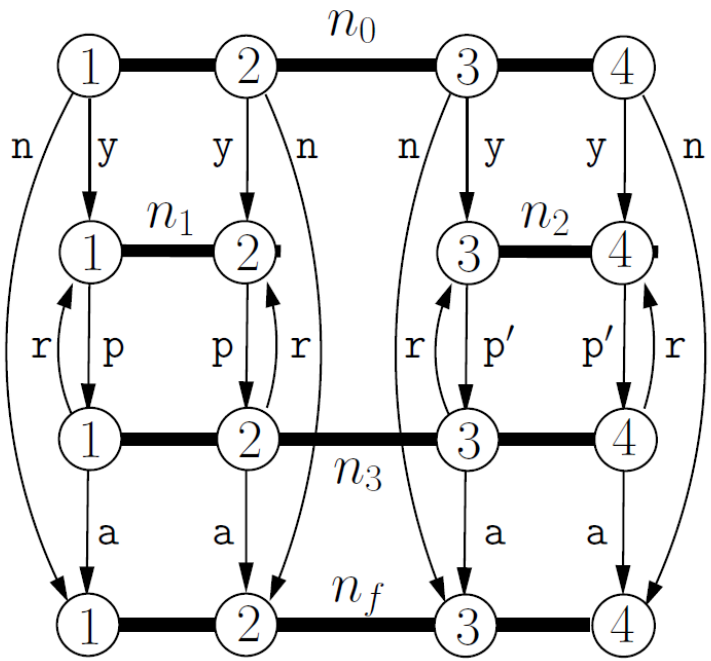


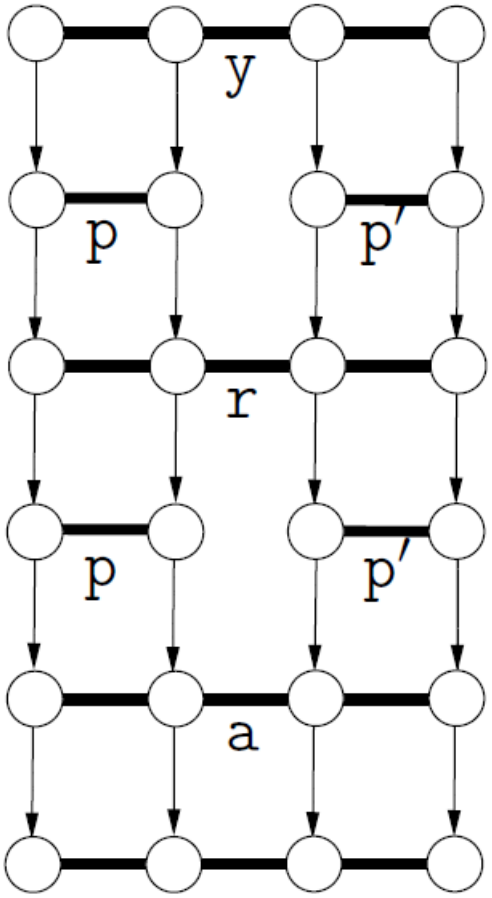
Semantics of negotiations: Mazurkiewicz traces



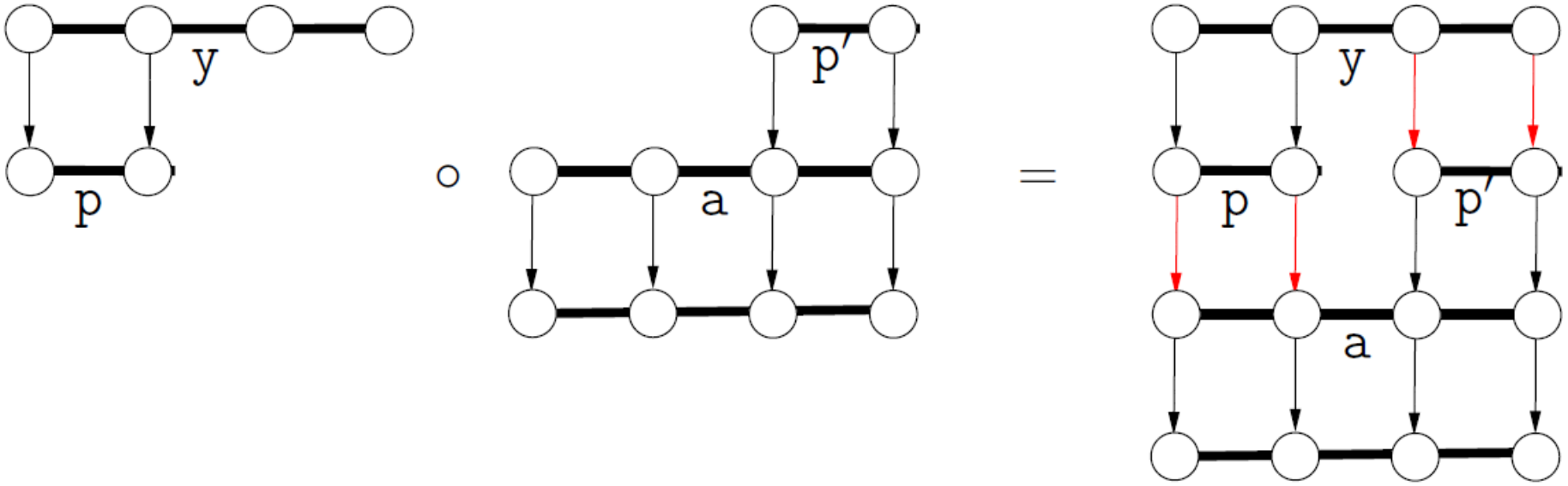


Semantics of negotiations: Mazurkiewicz traces





Mazurciewicz traces with concatenation



Independence of negotiation atoms (actually outcomes):

disjoint sets of participants

A grammar for negotiation programs

set of agents X

$$prog[X] ::= \epsilon$$

$$\mathbf{do}\ \{[]\ endalt[X]\}^{+}\ \{[]\ loopalt[X]\}^{*}\ \mathbf{od}$$

$$prog[Y] \circ prog[Z]$$

$$endalt[X] ::= name[X] : prog[X'] \ \mathbf{end}$$

$$loopalt[X] ::= name[X] : prog[X'] \ \mathbf{loop}$$

$$name[X] ::= element\ of\ \mathcal{R}_X$$

Y U Z = X

concatenation

for each X a name space

X‘ is a subset of X

Semantics of negotiation programs

the empty negotiation

$$[\![\epsilon]\!] = \{[\epsilon]\}$$

$$[\![\mathbf{do} \bigsqcap_{i=0}^{k} E_X^i \bigsqcap_{j=1}^{m} L_X^j \mathbf{od}]\!] = \left(\bigcup_{j=1}^{m} [\![L_X^j]\!]\right)^* \cdot \left(\bigcup_{i=0}^{k} [\![E_X^i]\!]\right)$$

$$[\![a : P_{X'}]\!] = \{[a]\} \cdot [\![P_{X'}]\!]$$

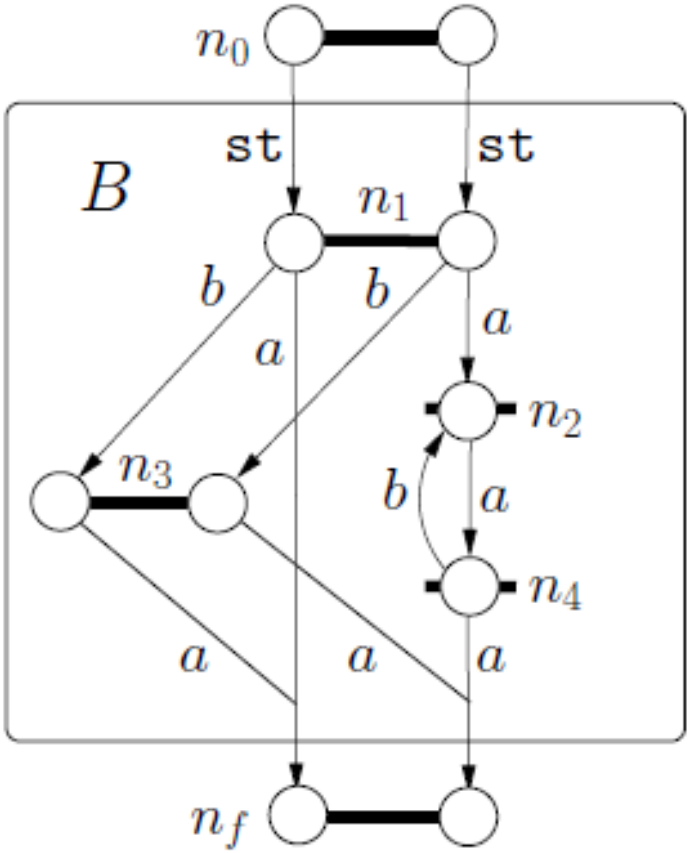
$$[\![P_Y \circ P'_Z]\!] = [\![P_Y]\!] \cdot [\![P'_Z]\!]$$

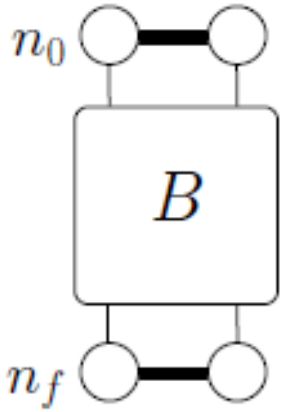
trace concatenation

end-alternative

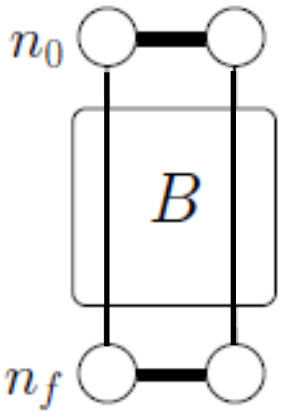
loop-alternative

Boxes

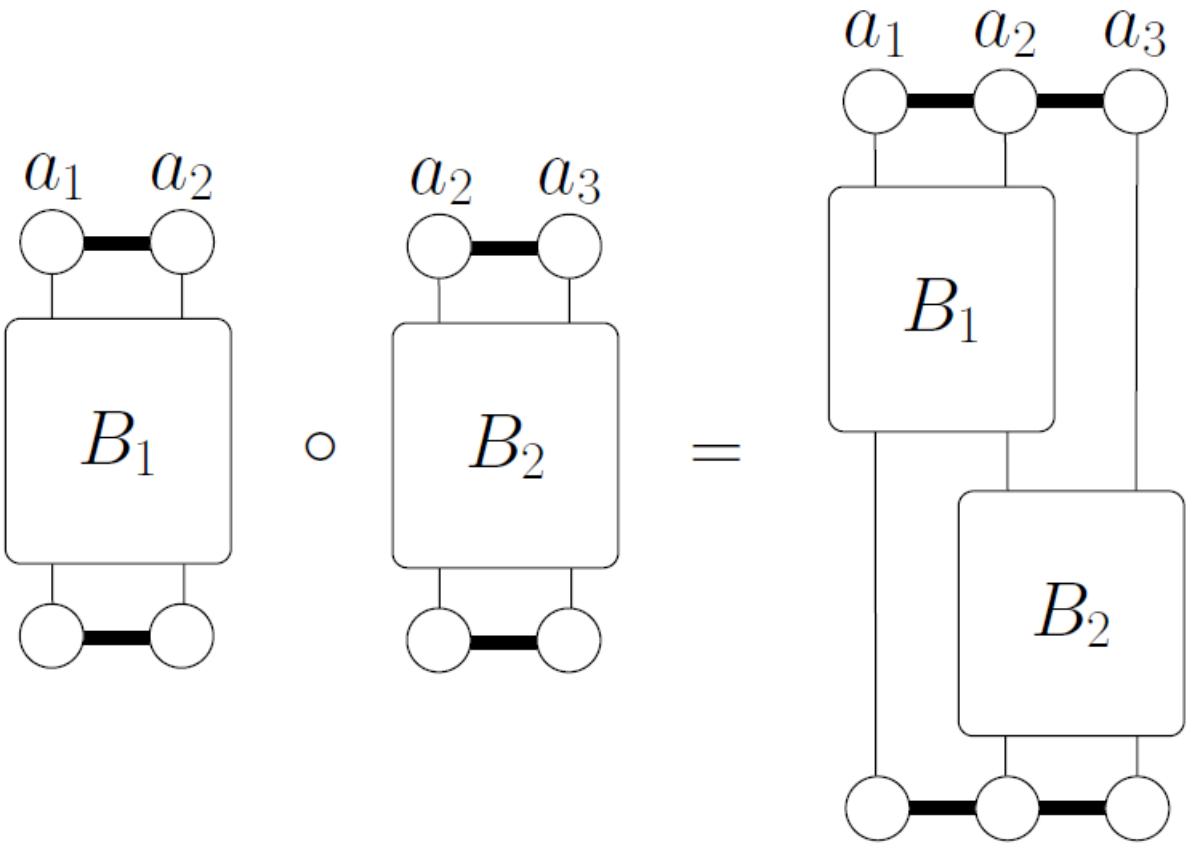




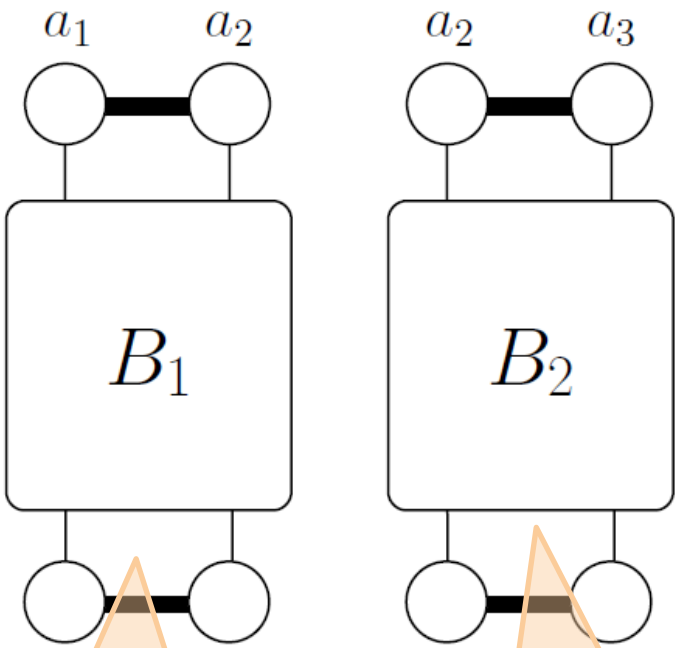
Box for ε



Conatenation of boxes

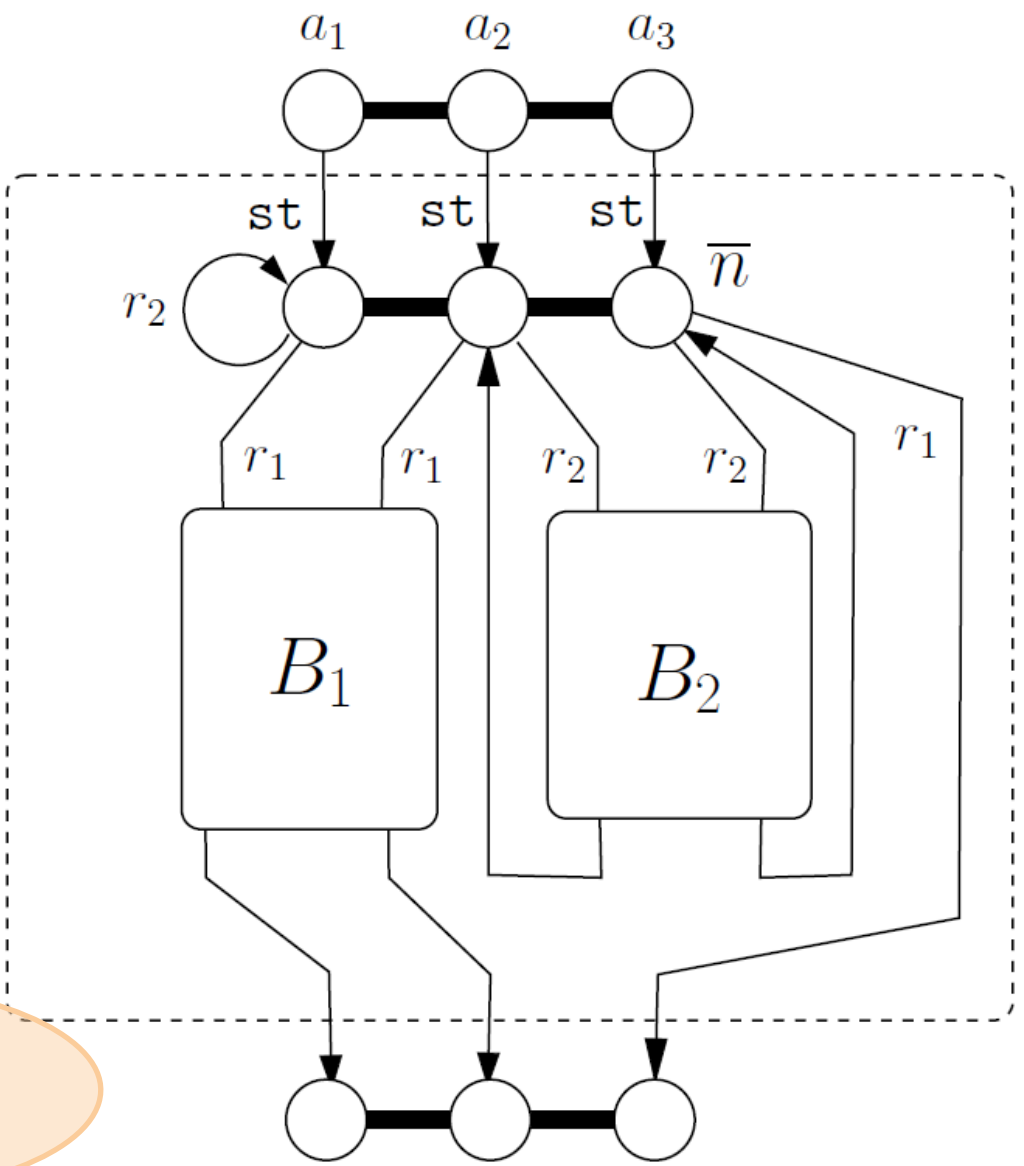


The alternative operator



end-alternative

loop-alternative



Equivalence result

equivalence:
same Mazurciewicz traces

Theorem:

- for each negotiation program there is an equivalent sound and deterministic negotiation which has almost the same size.
- for each sound and deterministic negotiation there is an equivalent negotiation program with the same set of agents.

proof based on reduction results

A concrete progamming language

$$
\begin{aligned}
prog[X] ::= \ & \mathbf{skip} \\
& comm[X] \\
& \mathbf{do}\ \{[]\ endgc[X]\}^{+}\ \{[]\ loopgc[X]\}^{*}\ \mathbf{od} \\
& prog[Y]\ \circ\ prog[Z] \\
endgc[X] ::= \ & guard[X] : prog[X']\ \mathbf{end} \\
loopgc[X] ::= \ & guard[X] : prog[X']\ \mathbf{loop} \\
guard[X] ::= \ & \textit{Guard over the variables of the agents of } X \\
comm[X] ::= \ & \textit{Command over the variables of the agents of } X
\end{aligned}
$$

(where neither guards nor commands have to use all variables of the respective sets X of agents).

concrete vs abstract

concrete:

$$\begin{array}{l}\textbf{agent } a \;\; \textbf{var } x = 1\text{: int} \\ \textbf{do } [] \; \neg(x \geq 1) \; : x = x - 1 \textbf{ loop} \\ \qquad [] \; (x < 1) \; : \; \textbf{skip end} \\ \textbf{od}\end{array}$$

abstract:

$$\begin{array}{l}\textbf{agent } a \\ \textbf{actions } \mathsf{a}, \mathsf{b}, \mathsf{c} : \{a\} \\ \textbf{do } [] \; \neg\mathsf{a} \; : \mathsf{b} \textbf{ loop} \\ \qquad [] \; \mathsf{c} \; : \; \textbf{end} \\ \textbf{od}\end{array}$$

a final example

agent a_1 **var** x_1 **:int**

...

agent a_4 **var** x_4 **:int**

1 **do** [] $\neg(x_1 = x_2 = x_3 = x_4)$:

2 $\{x_1, x_2 := f(x_1, x_2) \parallel x_3, x_4 := g(x_3, x_4)\} \circ$

3 **do** [] $(x_2 = x_3)$: **end**

4 [] $(x_2 \neq x_3)$:

5 $x_2, x_3 = h(x_2, x_3) \circ$

6 $\{x_1, x_2 := f(x_1, x_2) \parallel x_3, x_4 := g(x_3, x_4)\}$

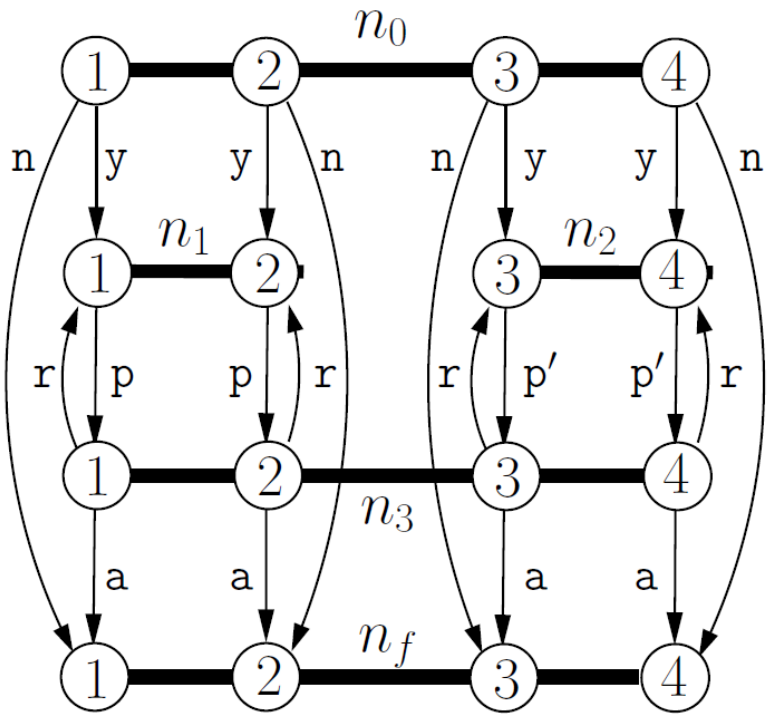
7 **end**

8 **od** **loop**

9 [] $(x_1 = x_2 = x_3 = x_4)$: **end**

10 **od**

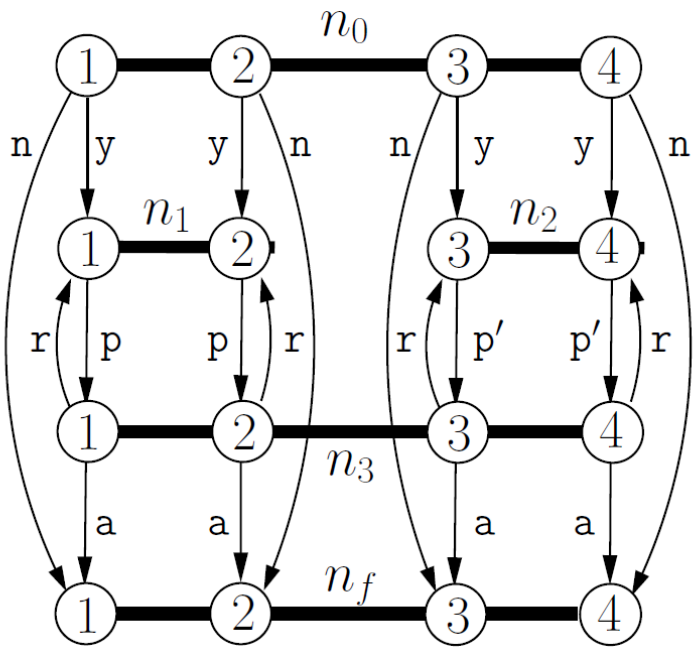
a final example



a final example

agent a_1 **var** x_1 **:int**
...
agent a_4 **var** x_4 **:int**

```
1   do [] ¬(x1 = x2 = x3 = x4) :
2         {x1, x2 := f(x1, x2)  ||  x3, x4 := g(x3, x4)} ◦
3         do [] (x2 = x3) : end
4            [] (x2 ≠ x3) :
5               x2, x3 = h(x2, x3) ◦
6               {x1, x2 := f(x1, x2) || x3, x4 := g(x3, x4
7               end
8          od  loop
9       [] (x1 = x2 = x3 = x4) : end
10  od
```



On Negotiation as Concurrency Primitive

Javier Esparza, Techn. Univ. München (D)

Jörg Desel, FernUniversität in Hagen (D)