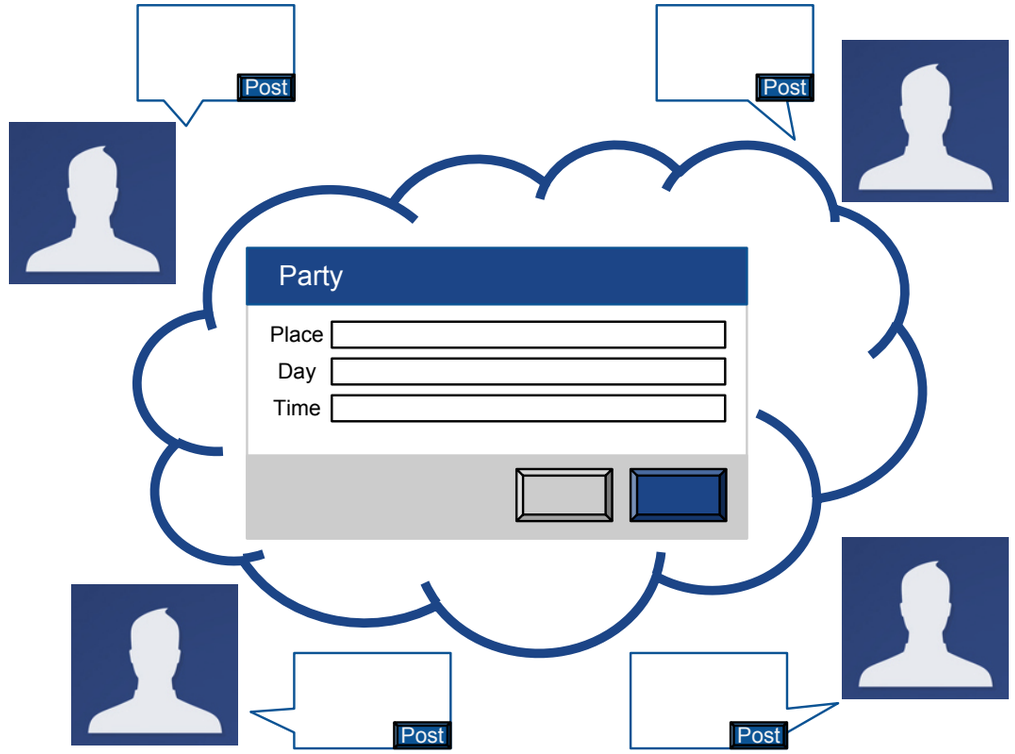
Bisimulation Techniques and Algorithms for Concurrent Constraint Programming

Andres Aristizabal's Ph.D Defense.
Supervisors: C. Palamidessi and F. Valencia.

DGA/CNRS and LIX, École Polytechnique Paris.
17th of October 2012

Our Motivation

- Advent of social networks where users post and share partial information.
  - Example: Facebook. Announce and ask information to friends via the public medium.



Our Motivation

- Relation with the DGA priorities:
  - Agent-based modeling or multi-agent system.
    - **Complex Systems:** Systems of systems.
- Importance of analysing the behavior of agents in a concurrent system.

Our Goal

- Choose a well-established formalism able to model concurrent systems where agents post and query partial information via a global store. e.g. social networks.
- Study the chosen model to understand the theory behind it.
- Aim at developing a new approach to labeled semantics and equivalences for the aforementioned formalism.

Our Approach

- Concurrent Constraint Programming (ccp)

  [SaraswatRinardPanangaden91]:

  - Not concerned with point-to-point channel communication but rather about systems of agents posting and querying information in some medium.
  - A mature formalism for concurrency with strong ties to logic.
  - Its agents can be seen both as processes and logic formulae.

Our Approach

- Bisimulation:
  - Central co-inductive technique to verify behavioral equivalences between programs in concurrency. [Milner80, Milner99].
  - A notion far too little considered in ccp. [SaraswatRinard90, MendlerPanangadenScottSeely95].
  - Definition for ccp over-discriminative. [SaraswatRinard90].
- Algorithms to verify bisimilarity for ccp:
  - Adapt and implement the Partition Refinement Algorithm [KanellakisSmolka83] to calculate the introduced notions of bisimilarity for ccp.

Our Contributions

- The thesis of this dissertation is that bisimulation is an adequate technique for the analysis and verification of programs in ccp.
- A sound and complete labeled transition semantics.
- A novel notion of labeled bisimilarity for ccp by building upon the works in [SaraswatRinard90, BonchiGadducciMonreale09].

Our Contributions

- A strong correspondence with existing ccp notions by providing a fully-abstract characterization of a standard observable behaviour for infinite ccp processes: The limits of fair computations.
- A fully-abstract correspondence with the closure operator denotational semantics of ccp.
- An algorithm to verify our notion of strong bisimilarity for ccp based on the works in [KanellakisSmolka83, BonchiMontanari09].

Our Contributions

- To the best of our knowledge, the first algorithm for the automatic verification of a ccp program equivalence.
- A proof that the relation given by the standard reduction from weak to strong bisimilarity is not complete for ccp.
- A complete new relation for ccp, which is finitely branching obtained by a different saturation mechanism.

Our Contributions

- The ccp Partition Refinement algorithm computes the weak bisimilarity for ccp by using the new relation obtained by the new saturation method.
- A correspondence between strong bisimilarity and irredundant bisimilarity.
- A correspondence between weak bisimilarity and irredundant bisimilarity over the new relation.

The Rest of this Talk: Outline

Outline

Constraint System

- A constraint system **C** is a complete algebraic lattice $(Con, Con_0, \sqsubseteq, \sqcup, true, false)$ where Con is a partially ordered set w.r.t. $\sqsubseteq$. Where:
  - Con_0 is the subset of *finite* elements of Con, $\sqcup$ is the lub operation, and *true*, *false* are the least and greatest elements of Con, respectively.
- Intuitively, in the particular logic case:
  - $c \sqsubseteq d$ means that d entails c (i.e., $d \vdash c$).
  - $c \sqcup d$ means c in conjunction with d (i.e., $c \wedge d$).

Example: The Herbrand Constraint System (cs)

- $\mathcal{L}$ is a first-order alphabet with variables $x, y, \ldots$ function symbols and equality $=$.
- Captures syntactic equality between terms $t, t' \ldots$ built from $\mathcal{L}$.
- Constraints are sets of equalities over the terms of $\mathcal{L}$.
- The relation $c \sqsubseteq d$ holds if the equalities in c follow from those in d.

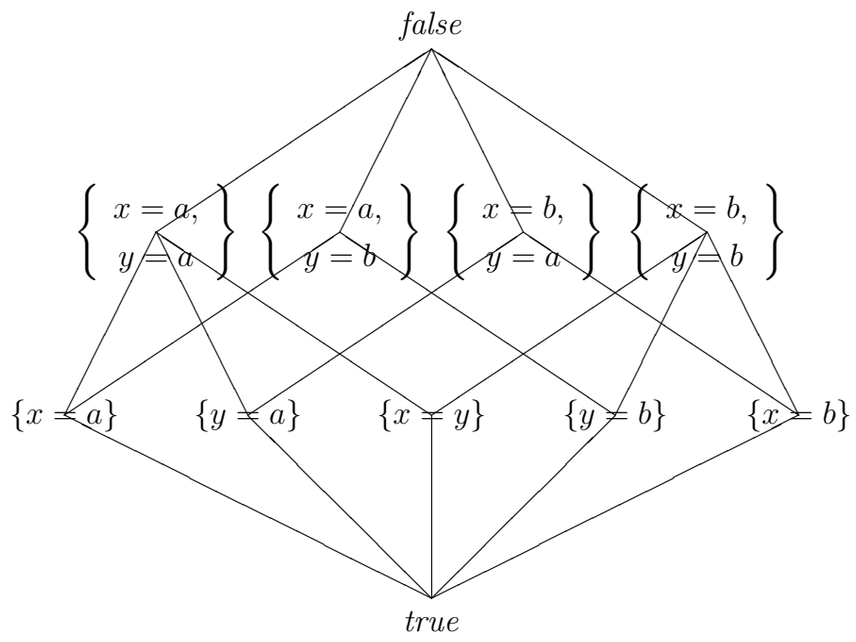
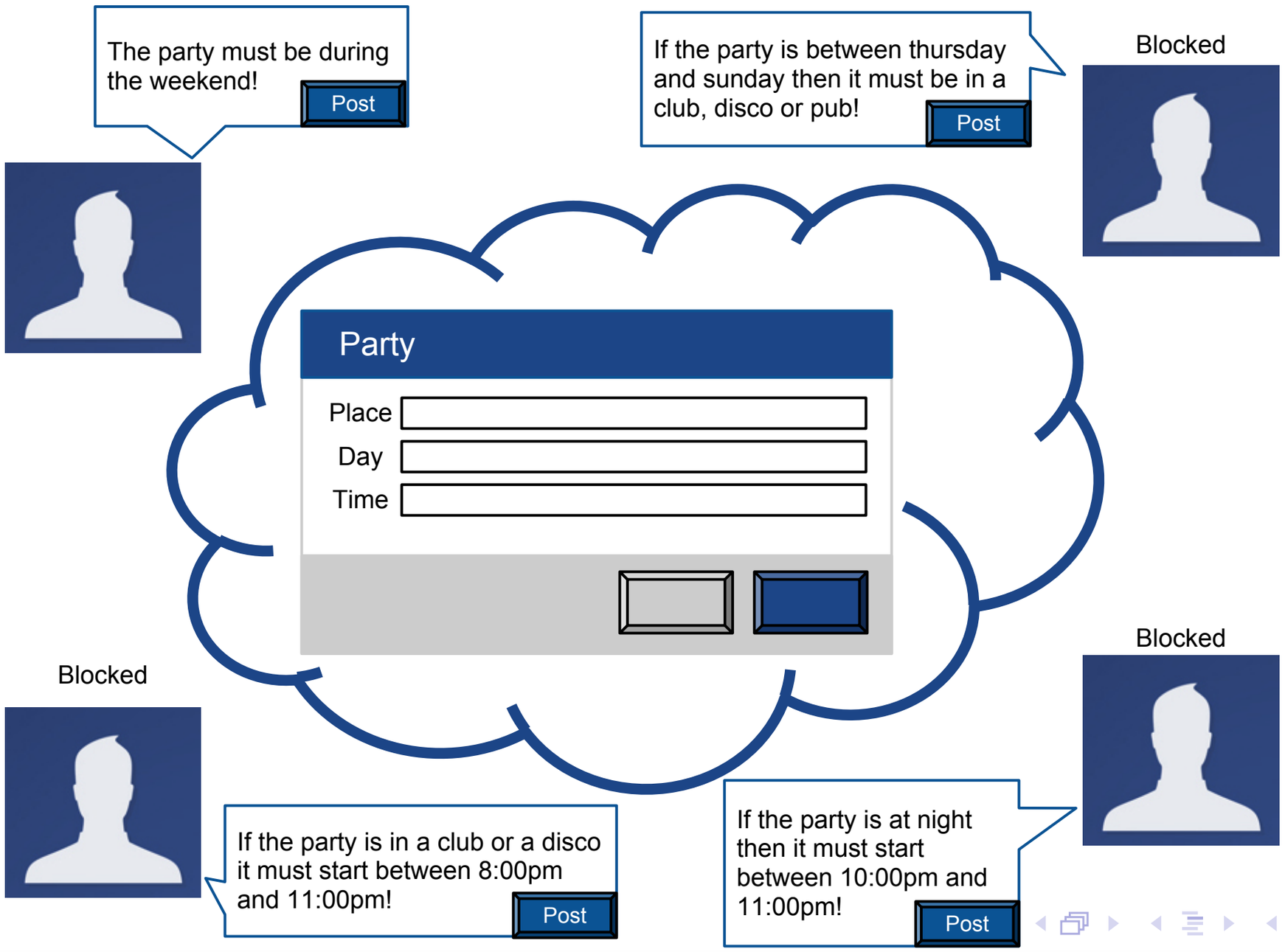
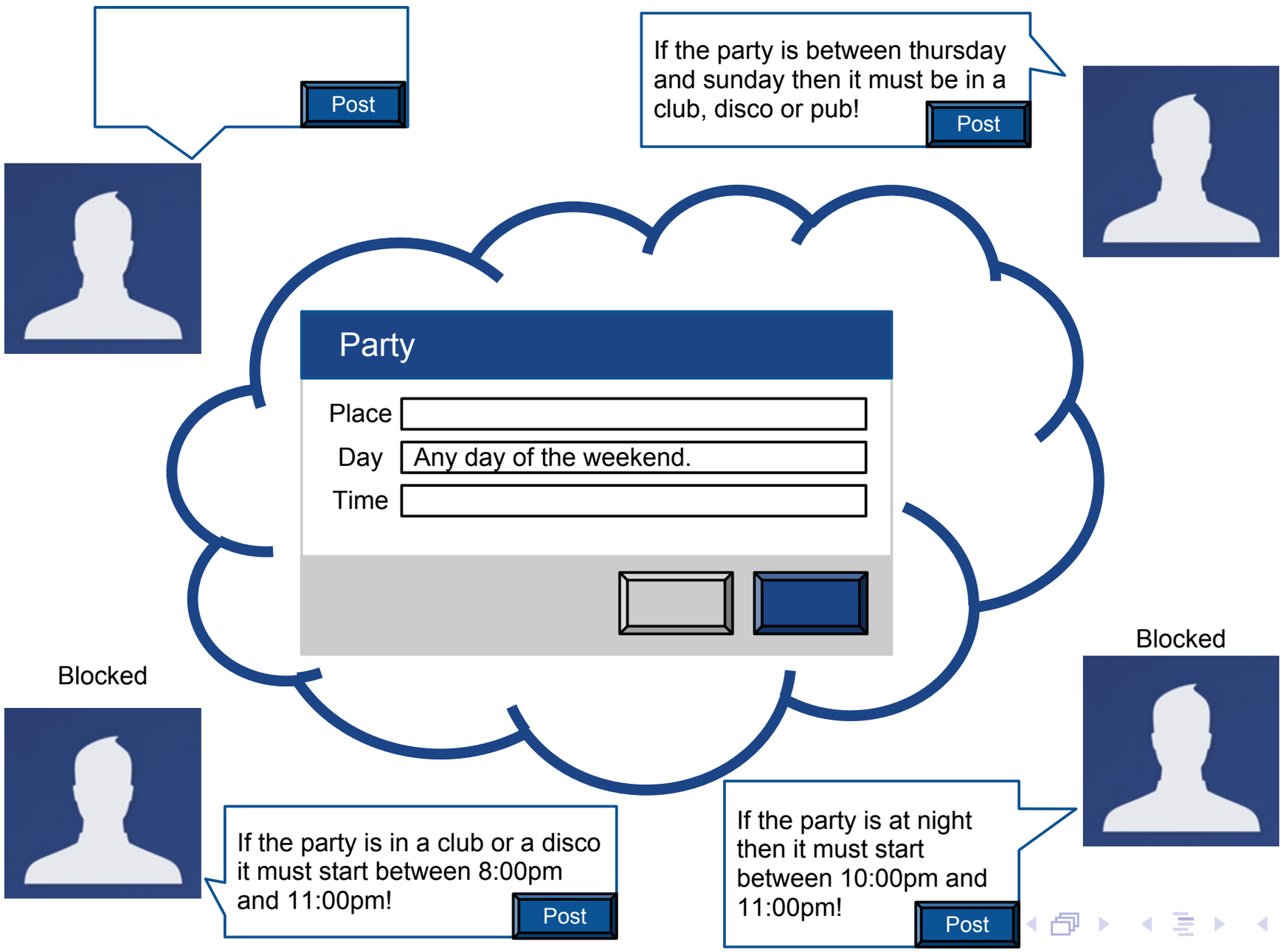


Figure 2.1: The Herbrand constraint lattice for x, y, a, b.

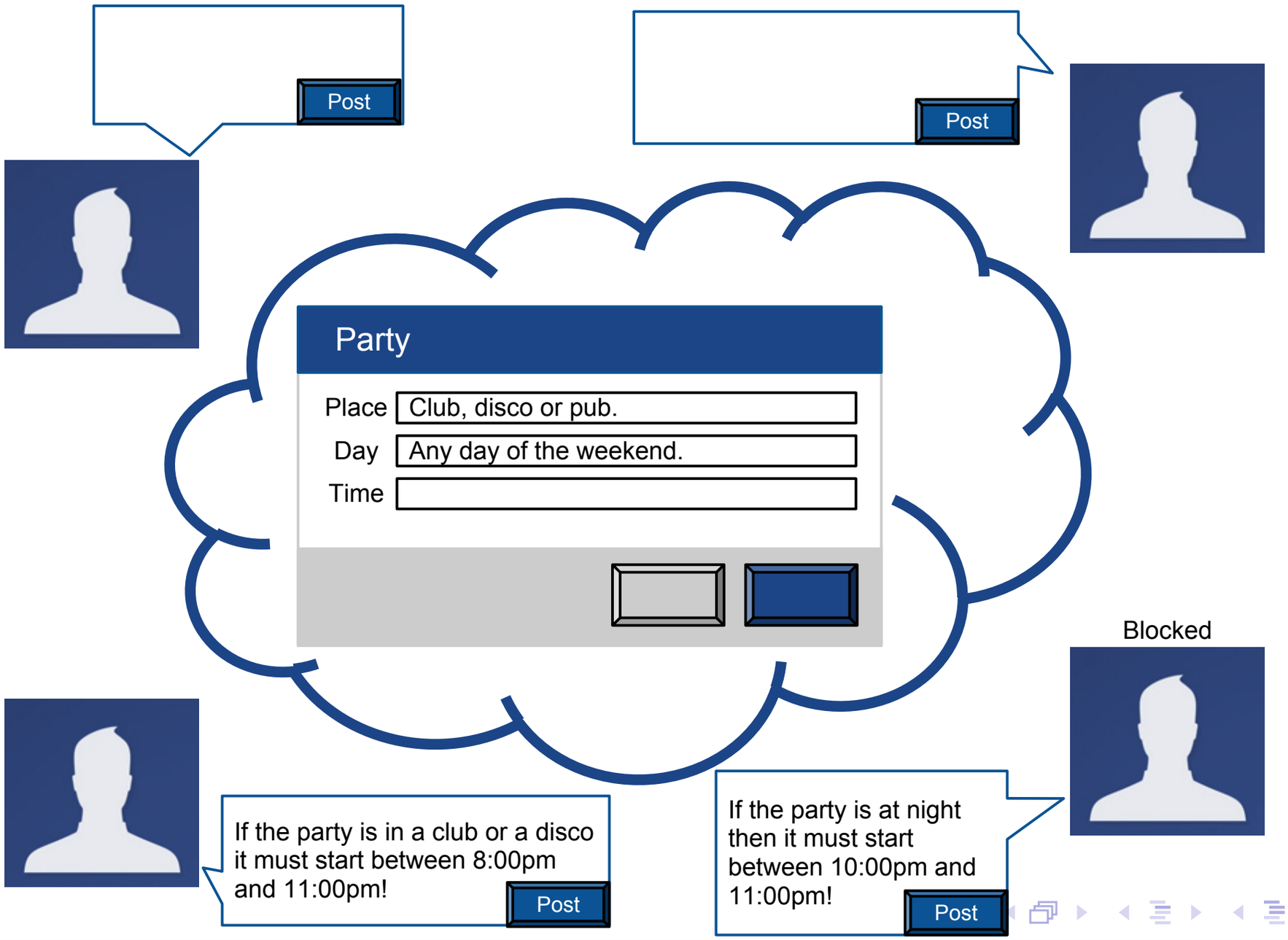
Concurrent Constraint Programming



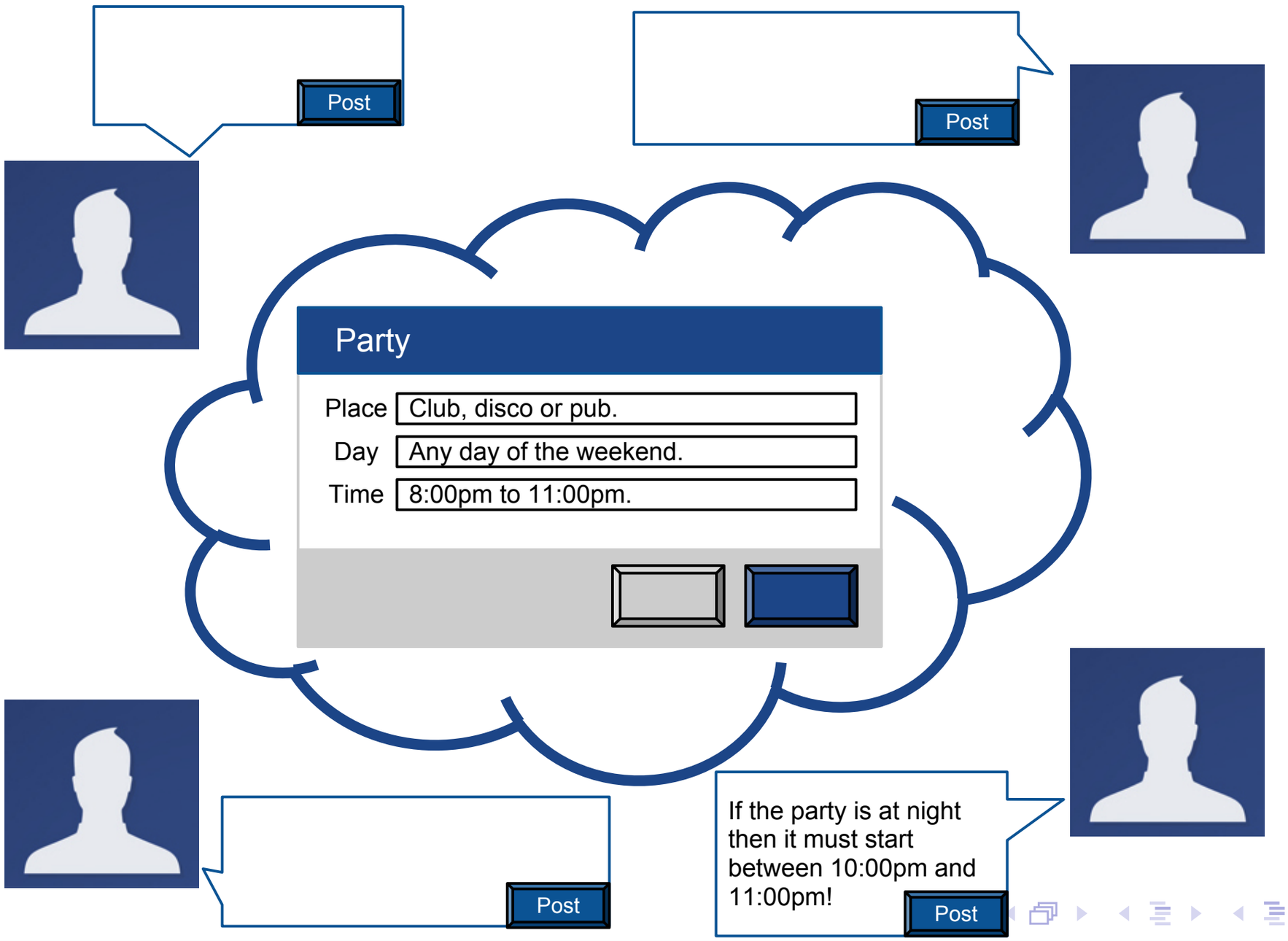
Concurrent Constraint Programming



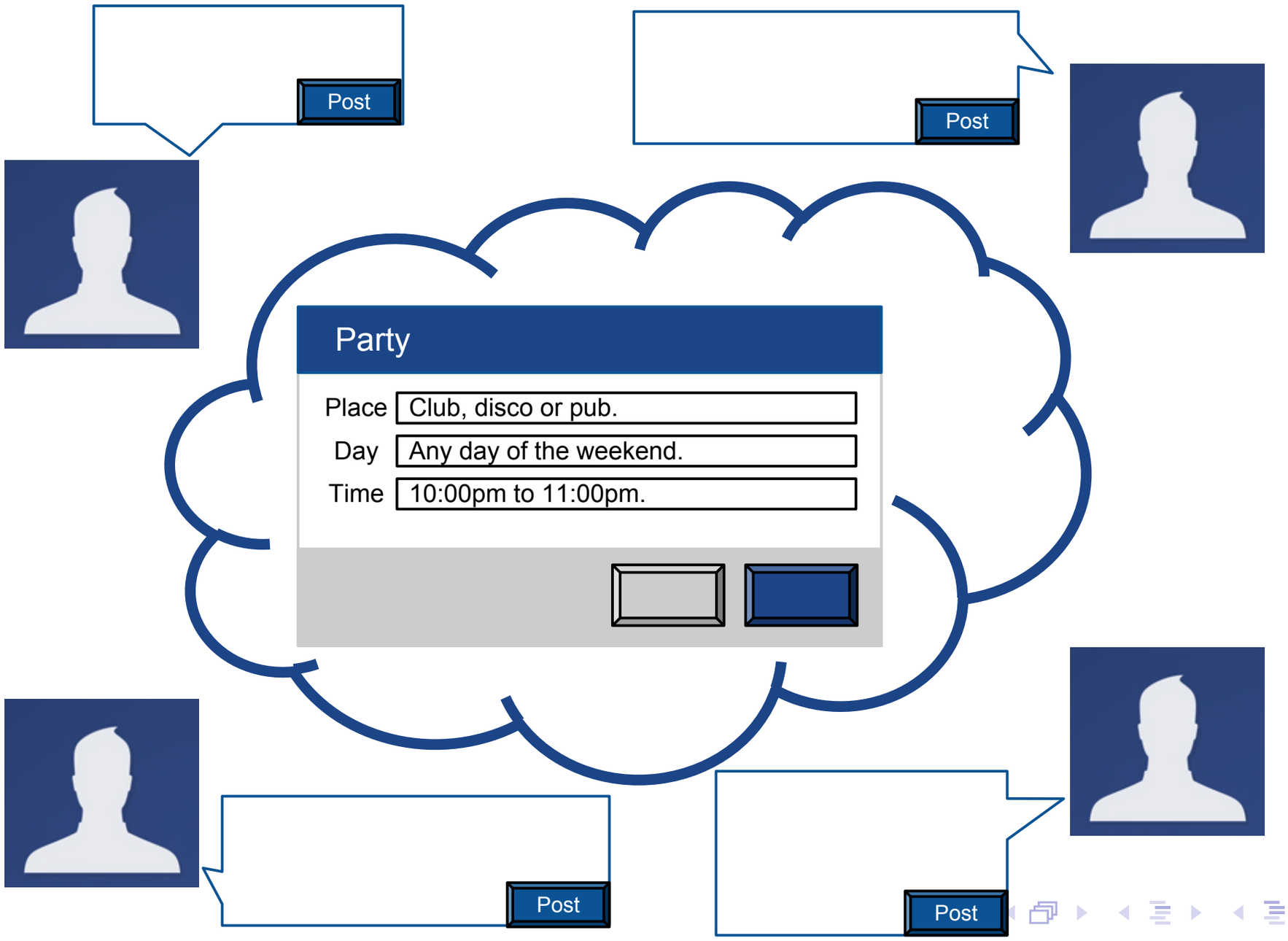
Concurrent Constraint Programming



Concurrent Constraint Programming



Concurrent Constraint Programming



Outline

Syntax

$$P, Q, \ldots ::= \mathbf{stop} \mid \mathbf{tell}(c) \mid \mathbf{ask}(c) \rightarrow P \mid P \parallel Q \mid \exists_x^e P \mid p(\vec{z})$$

- $\mathbf{tell}(c)$ adds the constraint c to the store,
- $\mathbf{ask}\ c \rightarrow P$ executes P iff the store entails c,
- $P \parallel Q$ means P and Q execute in parallel. They may interact.
- $\exists_x^e P$ represents x hidden in P with the local store e.

Outline

Operational Semantics

Reduction Semantics

Semantics via reduction relation between configurations $\gamma = \langle P, c \rangle$ where P is a process and c is the store representing the partial information currently known.

$$\langle \mathbf{tell}(c), d \rangle \longrightarrow \langle \mathbf{stop}, d \sqcup c \rangle \qquad \frac{c \sqsubseteq d}{\langle \mathbf{ask}\ (c)\ \rightarrow\ P, d \rangle \longrightarrow \langle P, d \rangle}$$

$$\frac{\langle P, d \rangle \longrightarrow \langle P', d' \rangle}{\langle P \parallel Q, d \rangle \longrightarrow \langle P' \parallel Q, d' \rangle} \qquad \frac{\langle P, e \sqcup \exists_x d \rangle \longrightarrow \langle P', e' \sqcup \exists_x d \rangle}{\langle \exists_x^e P, d \rangle \longrightarrow \langle \exists_x^{e'} P', d \sqcup \exists_x e' \rangle}$$

Outline

Observational Equivalence

- A process P is active in a transition if it generates such transition.
- A process P is enabled in γ if there exists a γ' such that P is active in $\gamma \longrightarrow \gamma'$.

Observational Equivalence

- A possibly infinite computation
  $\xi = \gamma_0 \longrightarrow \gamma_1 \longrightarrow \ldots \longrightarrow \gamma_n \ldots$ is fair iff, for each process enabled in some γ_i there exists $j \geq i$ such that the process is active in a transition $t = \gamma_i \longrightarrow \gamma_j$.
- The result of ξ is the limit $\bigsqcup_i d_i$ where d_i is the store of γ_i.
- $P \sim_o Q$ iff initiated in the same store, the results of their fair computations are the same.

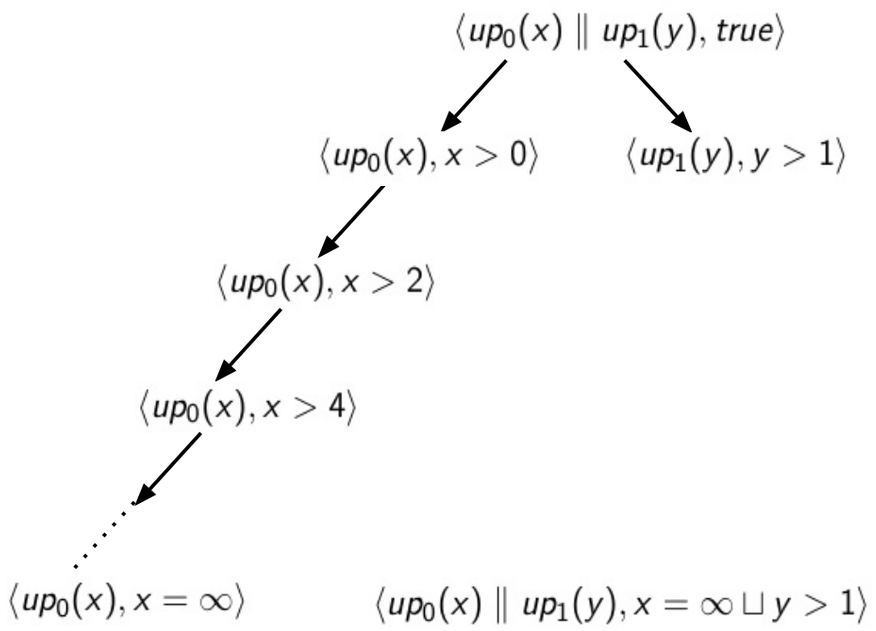
Example: $\langle up_0(x) \parallel up_1(y), true\rangle \sim_o \langle fairup(x,y,z), true\rangle$

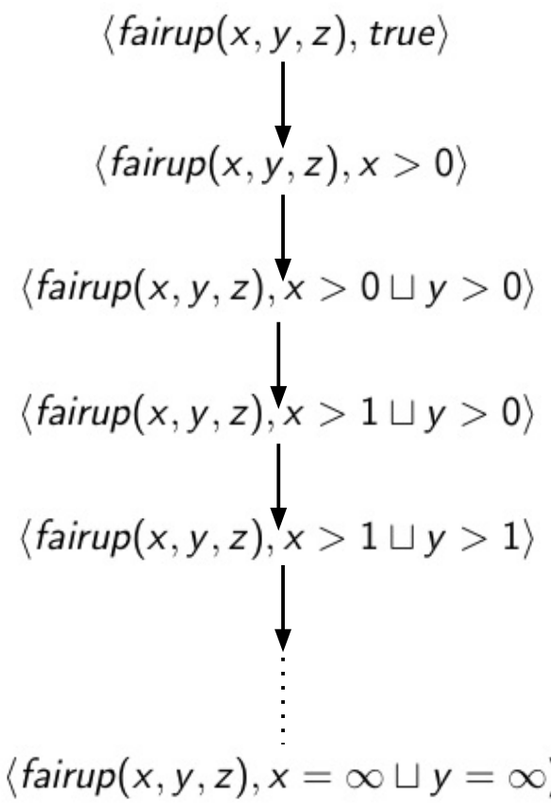
$$up_n(x) \stackrel{\text{def}}{=} \exists_y(\mathbf{tell}(y = n) \mid \mathbf{ask}\ (y = n) \ \rightarrow \ up(x,y))$$

$$up(x,y) \stackrel{\text{def}}{=} \exists_{y'}(\mathbf{tell}(y < x \wedge succ^2(y,y')) \mid \mathbf{ask}(y < x \wedge succ^2(y,y')) \rightarrow up(x,y'))$$

$$fairup(x,y,z) \stackrel{\text{def}}{=}$$

$$\exists_{z'}(\mathbf{tell}(z < x \wedge succ(z,z')) \mid \mathbf{ask}\ ((z < x) \wedge succ(z,z')) \ \rightarrow \ fairup(y,x,z')))$$





Outline

Barbs

- Barbed equivalences have become the standard behavioral equivalences for formalisms equipped with unlabeled reduction semantics.
- We have to fix a set of barbs i.e., basic observations on the states of processes.
- Barbs for ccp are constraints.
- A configuration $\langle P, d\rangle$ satisfies the barb c (written $\langle P, d\rangle \downarrow_c$) iff $c \sqsubseteq d$.
- Write $\gamma \Downarrow_c$ iff $\gamma \rightarrow^* \gamma' \downarrow_c$, where $\rightarrow^*$ is the transitive and reflexive closure of $\rightarrow$.

Saturated barbed bisimilarity

Definition

(Panangaden et al '95). A saturated barbed bisimulation is a symmetric relation $\mathcal{R}$ on configurations such that whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, d \rangle$ and $\gamma_2 = \langle Q, e \rangle$ implies that:

(i) if $\gamma_1 \downarrow_c$ then $\gamma_2 \downarrow_c$ and

(ii) if $\gamma_1 \rightarrow \gamma_1'$ then $\exists \gamma_2'$ s.t. $\gamma_2 \rightarrow \gamma_2'$ and $(\gamma_1', \gamma_2') \in \mathcal{R}$.

(iii) $\forall a : (\langle P, d \sqcup a \rangle, \langle Q, e \sqcup a \rangle) \in \mathcal{R}$.

Define $\gamma_1 \dot{\sim}_{sb} \gamma_2$ iff there exists a saturated barbed bisimulation $\mathcal{R}$ such that $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Write $P \dot{\sim}_{sb} Q$ iff $\langle P, true \rangle \dot{\sim}_{sb} \langle Q, true \rangle$.

- For weak version $\dot{\approx}_{sb}$ replace $\rightarrow$ with $\rightarrow^*$ and $\downarrow_c$ with $\Downarrow_c$.

Outline

Labeled Semantics

- The saturated version is a little unsatisfactory because of the universal quantification.
- Define a LTS where a labeled transition

$$\gamma \xrightarrow{\alpha} \gamma'$$

where the constraint α represents some minimal information needed from the environment to evolve from γ into γ'.

Example

$\langle \textbf{ask}\ (x > 10) \rightarrow P, \mathit{true} \rangle \xrightarrow{\alpha} \gamma$ then $\alpha = x > 10$. Even if α could be $x > 20, x > 34, \ldots$ for the first configuration to evolve.

Labeled Semantics

$$\langle \mathbf{tell}(c), d\rangle \xrightarrow{\textit{true}} \langle \mathbf{stop}, d \sqcup c\rangle$$

$$\frac{\alpha \in \min\{a \in \mathit{Con}_0 \mid c \sqsubseteq d \sqcup a\}}{\langle \mathbf{ask}\ (c) \ \rightarrow \ P, d\rangle \xrightarrow{\alpha} \langle P, d \sqcup \alpha\rangle}$$

$$\frac{\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle}{\langle P \parallel Q, d\rangle \xrightarrow{\alpha} \langle P' \parallel Q, d'\rangle}$$

$$\frac{\langle P[z/x], e[z/x] \sqcup d\rangle \xrightarrow{\alpha} \langle P', e' \sqcup d \sqcup \alpha\rangle}{\langle \exists^{e}_{x} P, d\rangle \xrightarrow{\alpha} \langle \exists^{e'[x/z]}_{x} P'[x/z], \exists_x(e'[x/z]) \sqcup d \sqcup \alpha\rangle} \; x \notin \mathit{fv}(e'), z \notin \mathit{fv}(P) \cup \mathit{fv}(e \sqcup d \sqcup \alpha)$$

Syntactic bisimilarity

Definition

(SaraswatRinard '90). A syntactic bisimulation is a symmetric relation $\mathcal{R}$ on configurations such that whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ implies that:

(i) if $\gamma_1 \downarrow_c$ then $\gamma_2 \downarrow_c$,

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ then $\exists \gamma_2'$ such that $\gamma_2 \xrightarrow{\alpha} \gamma_2'$ and $(\gamma_1', \gamma_2') \in \mathcal{R}$.

Define $\gamma_1 \sim_S \gamma_2$ iff there exists a syntactic bisimulation $\mathcal{R}$ such that $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Write $P \sim_S Q$ iff $\langle P, \mathit{true} \rangle \sim_S \langle Q, \mathit{true} \rangle$.

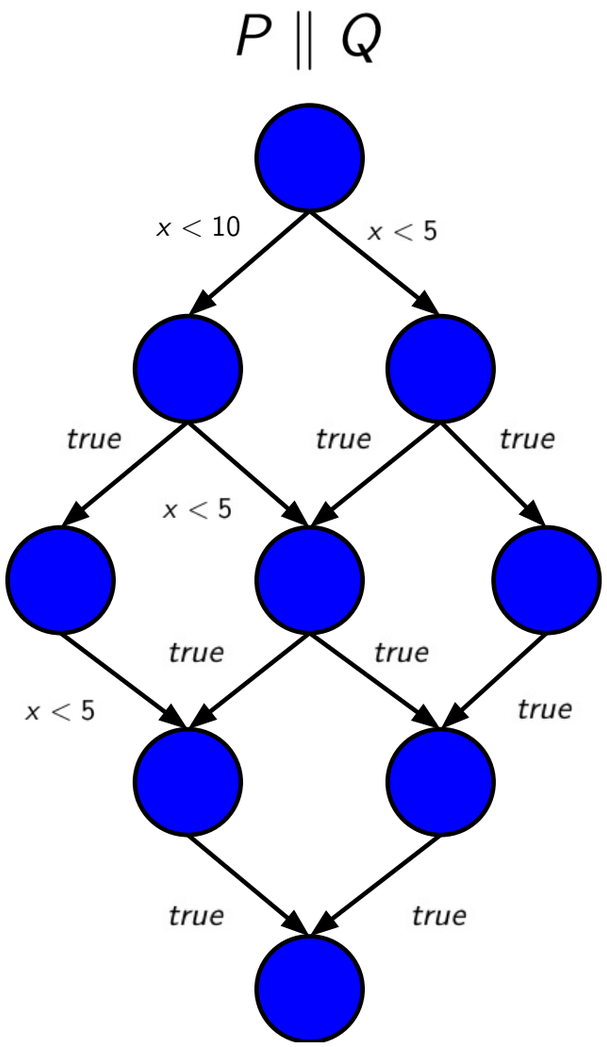
Example (over discriminating)

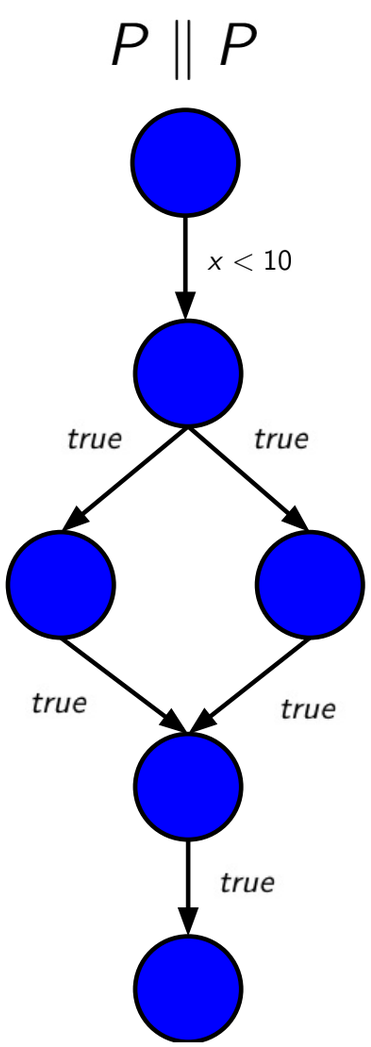
Let $P = \mathbf{ask}\ (x < 10) \ \rightarrow\ \mathbf{tell}(x < 5)$ and

$Q = \mathbf{ask}\ (x < 5) \ \rightarrow\ \mathbf{tell}(x < 5)$.

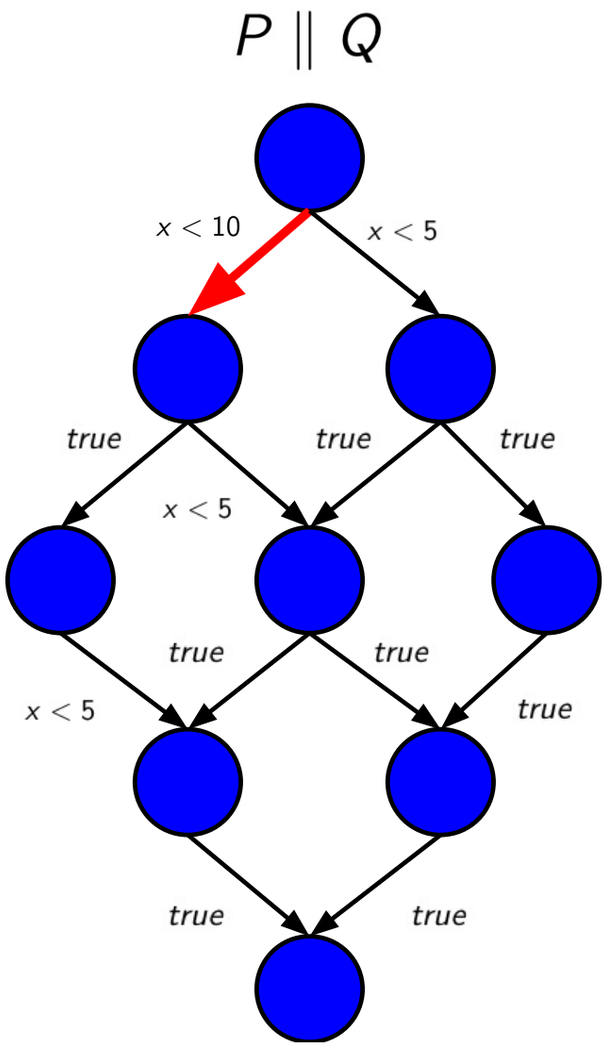
We have $\langle P \parallel Q, true\rangle \ \not\sim_S \langle P \parallel P, true\rangle$.

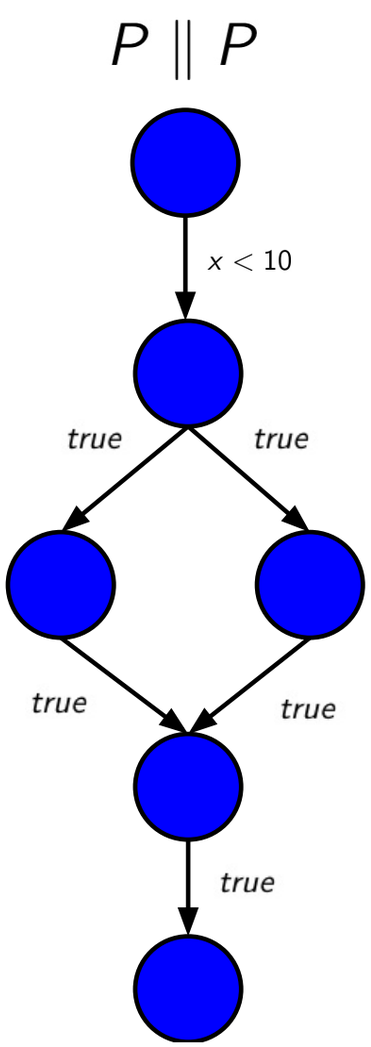
Example (over discriminating)



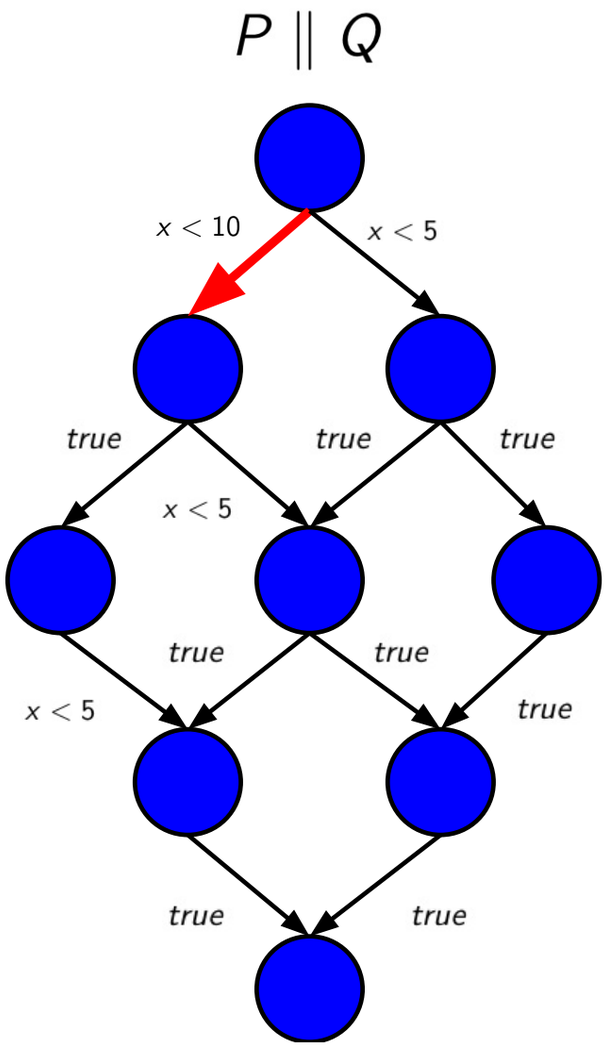


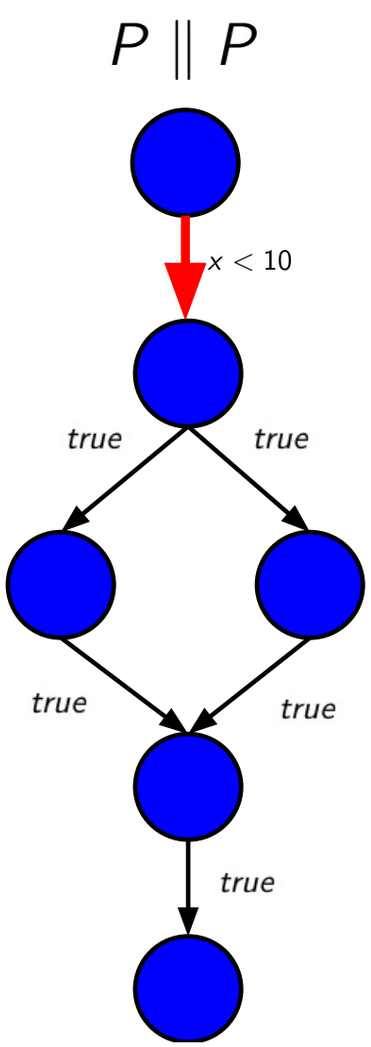
Example (over discriminating)



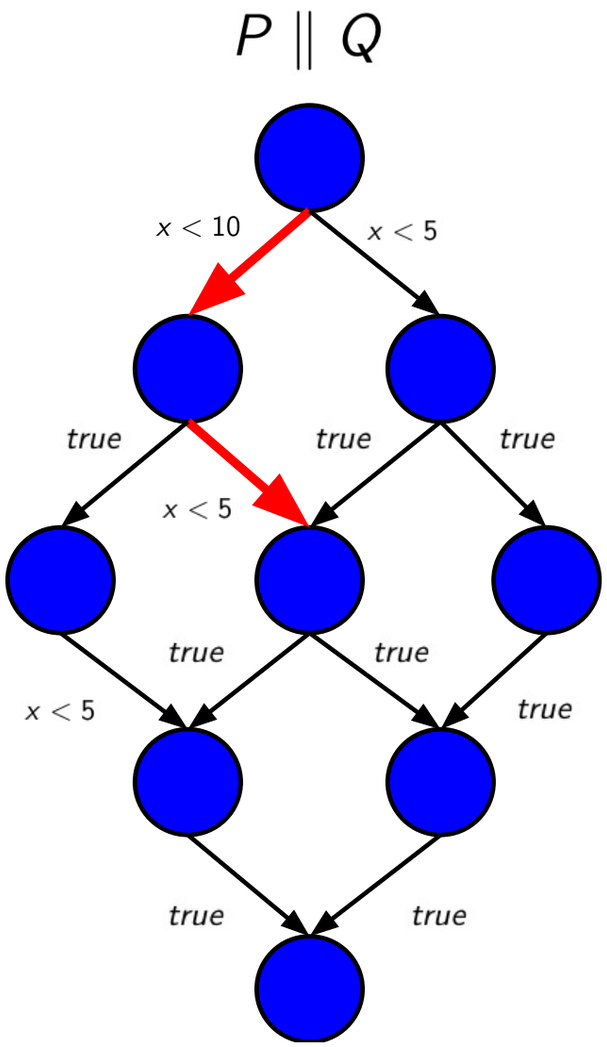


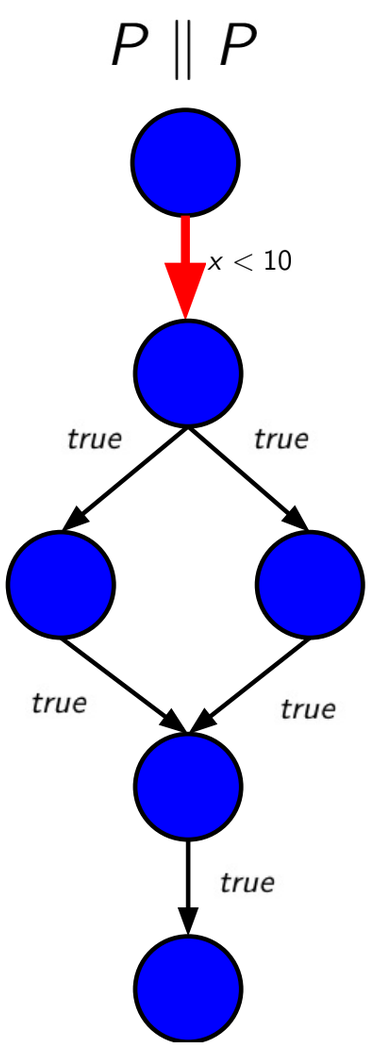
Example (over discriminating)



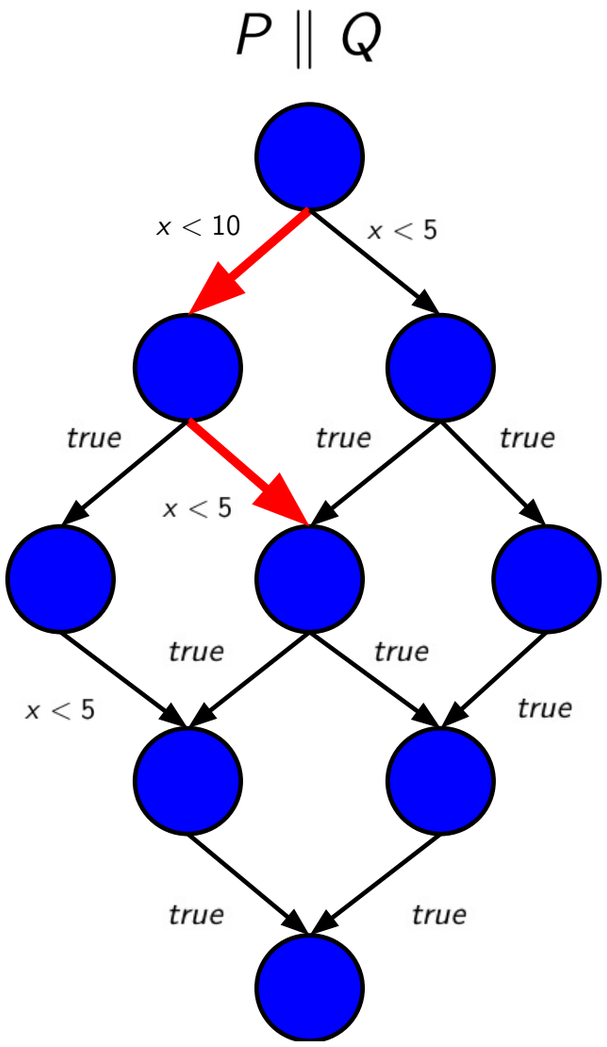


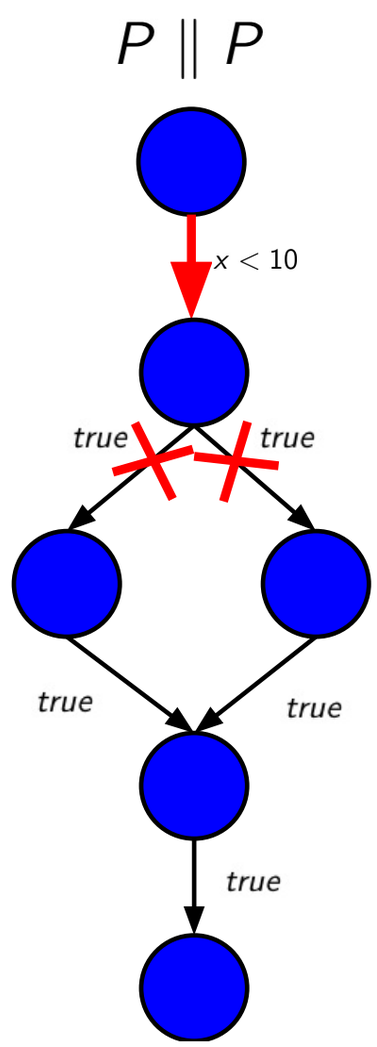
Example (over discriminating)



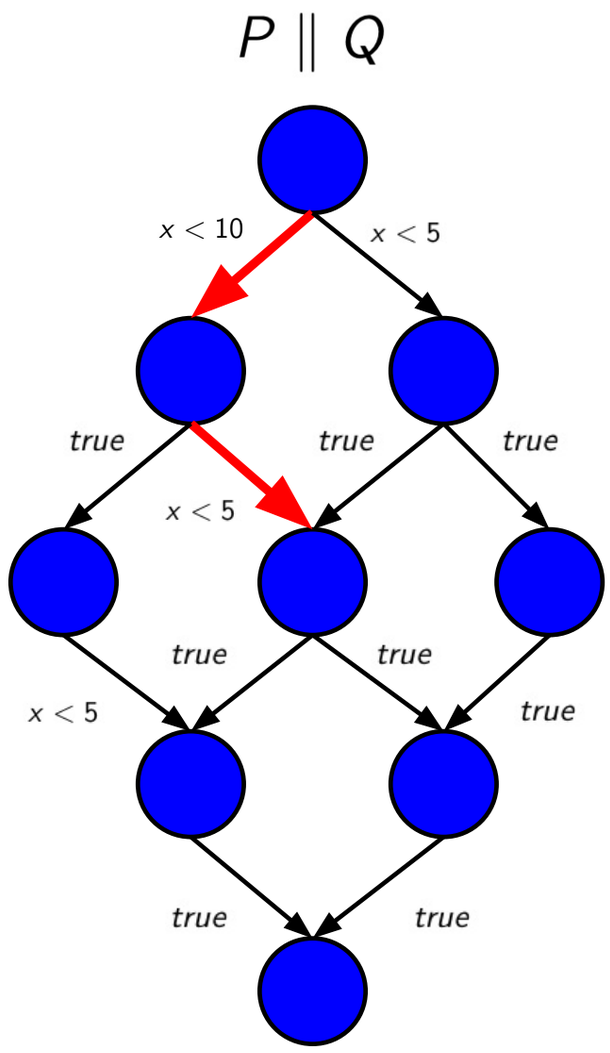


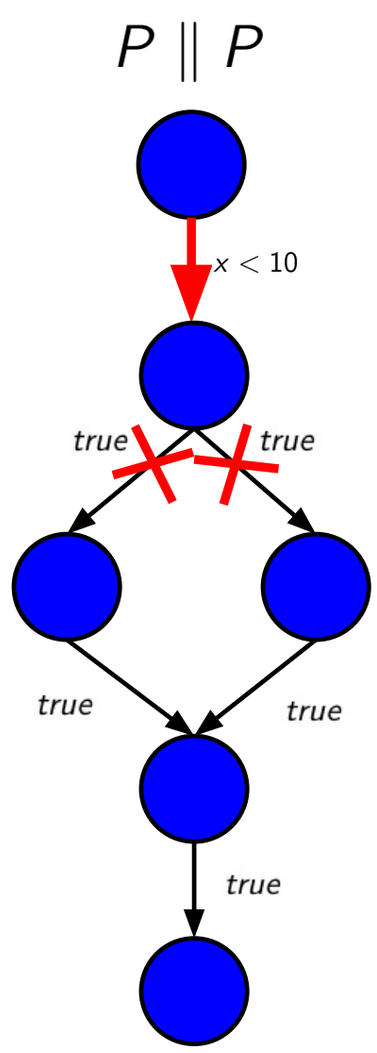
Example (over discriminating)





Example (over discriminating)





Nevertheless $\langle P \parallel Q, true \rangle \sim_o \langle P \parallel P, true \rangle$.

Strong bisimilarity

Definition

A strong bisimulation is a symmetric relation $\mathcal{R}$ on configurations such that whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, d \rangle$ and $\gamma_2 = \langle Q, e \rangle$ implies that:

(i) if $\gamma_1 \downarrow_c$ then $\gamma_2 \downarrow_c$ and

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ then $\exists \gamma_2'$ s.t. $\langle Q, e \sqcup \alpha \rangle \rightarrow \gamma_2'$ and $(\gamma_1', \gamma_2') \in \mathcal{R}$.

Define $\gamma_1 \dot{\sim} \gamma_2$, if there exists a strong bisimulation $\mathcal{R}$ such that $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Write $P \dot{\sim} Q$ iff $\langle P, \mathit{true} \rangle \dot{\sim} \langle Q, \mathit{true} \rangle$.

Correspondence results

Our notions are well-behaved w.r.t. barbed congruence.

Theorem

$\dot{\sim} = \dot{\sim}_{sb}$

Theorem

$\dot{\approx} = \dot{\approx}_{sb}$.

Correspondence results

A fully-abstract characterization.

Theorem

$P \dot{\approx}_{sb} Q$ *iff* $P \sim_o Q$.

Remark

Our characterization in terms of bisimilarity avoids complicated notions such as fairness and infinite elements.

Outline

Standard Partition Refinement

F is defined as follows: for all partitions $\mathcal{P}$, $P\,\mathbf{F}(\mathcal{P})\,Q$ iff

- if $P \xrightarrow{\lambda} P'$ then exists Q' s.t. $Q \xrightarrow{\lambda} Q'$ and $P'\,\mathcal{P}\,Q'$.

Algorithm 1 `Partition-Refinement(`IS`)`

Initialization

1. $IS^\star$ is the set of all processes reachable from IS,
2. $\mathcal{P}^0 := \{IS^\star\}$,

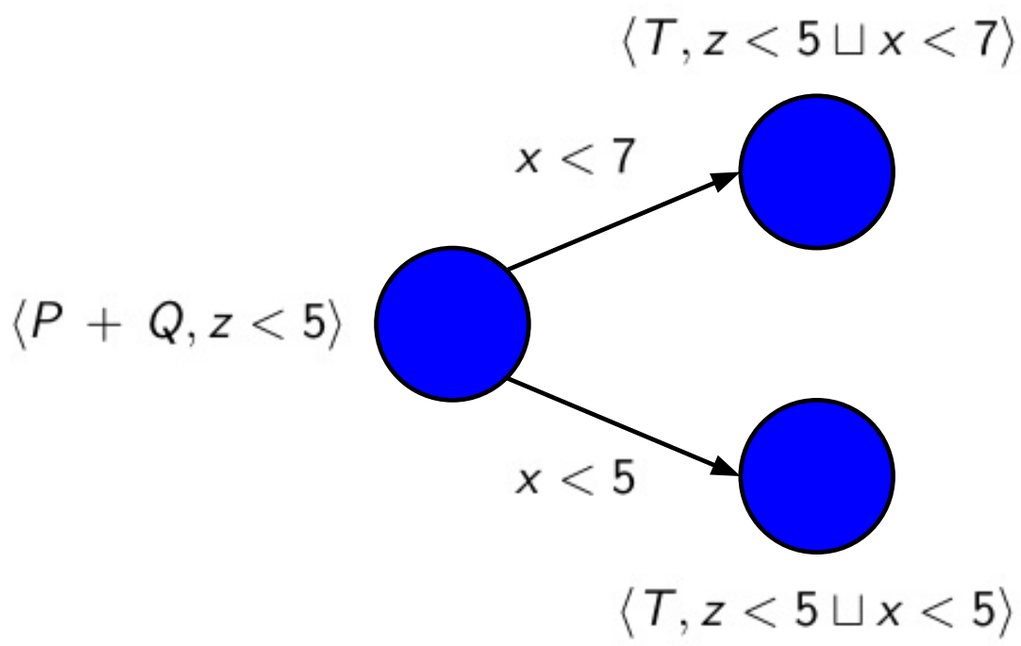
Iteration $\mathcal{P}^{n+1} := \mathbf{F}(\mathcal{P}^n)$,
Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

Why Standard Partition Refinement does not work? Syntactic bisimilarity is too fine-grained

$$T = \mathbf{tell}(true)$$

$$P = \mathbf{ask}\ (x < 7)\ \rightarrow\ T$$

$$Q = \mathbf{ask}\ (x < 5)\ \rightarrow\ T$$

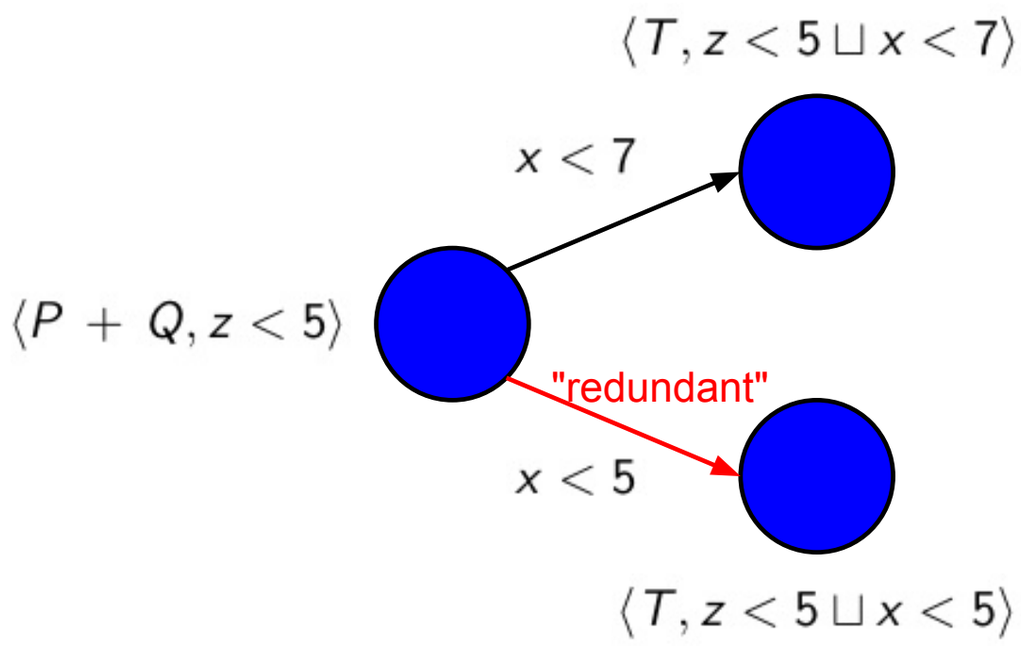


Why Standard Partition Refinement does not work? Syntactic bisimilarity is too fine-grained

$$T = \mathbf{tell}(true)$$

$$P = \mathbf{ask}\ (x < 7)\ \rightarrow\ T$$

$$Q = \mathbf{ask}\ (x < 5)\ \rightarrow\ T$$

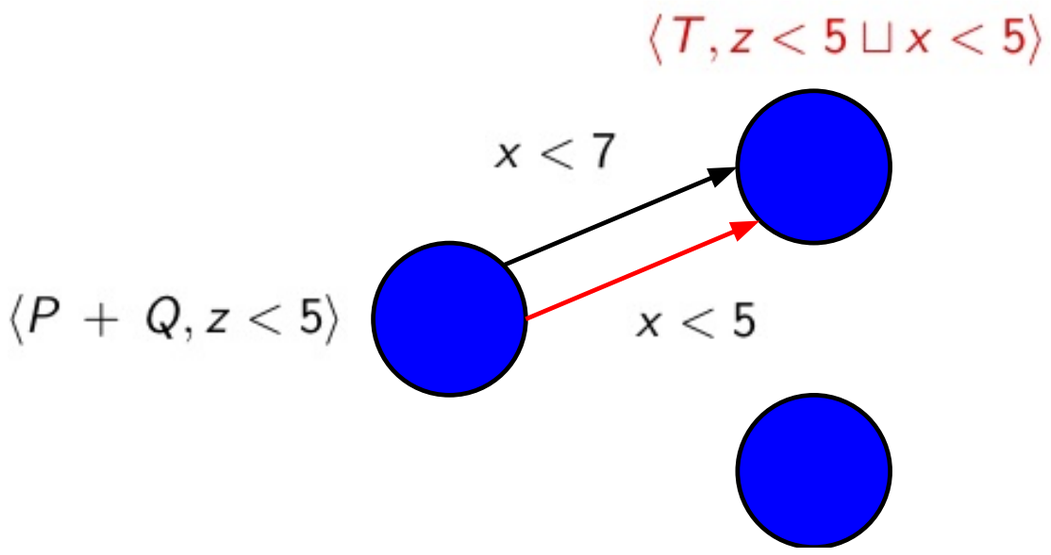


Why Standard Partition Refinement does not work? Syntactic bisimilarity is too fine-grained

$$T = \mathbf{tell}(\mathit{true})$$

$$P = \mathbf{ask}\ (x < 7)\ \rightarrow\ T$$

$$Q = \mathbf{ask}\ (x < 5)\ \rightarrow\ T$$



Outline

Derivation, domination and redundancy

Definition (Derivation $\vdash_D$)

We say that $\langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle$ derives $\langle P, c\rangle \xrightarrow{\beta} \langle P_2, c''\rangle$, written $\langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle \vdash_D \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c''\rangle$, iff there exists e s.t. the following conditions hold:

(i) $\beta = \alpha \sqcup e$ (ii) $c'' = c' \sqcup e$

Derivation, domination and redundancy

Definition (Domination $\succ_D$)

We say that $\langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle$ dominates $\langle P, c\rangle \xrightarrow{\beta} \langle P_2, c''\rangle$, written $\langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle \succ_D \langle P, c\rangle \xrightarrow{\beta} \langle P_2, c''\rangle$, iff there exists e s.t. the following conditions hold:

(i) $\beta = \alpha \sqcup e$ (ii) $c'' = c' \sqcup e$ (iii) $\alpha \neq \beta$

Derivation, domination and redundancy

Definition (Redundant transition)

Let $\mathcal{R}$ be a relation and $t = \langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle$ and $t' = \langle P, c\rangle \xrightarrow{\beta} \langle P_2, c''\rangle$ be two transitions. $t \succ_{\mathcal{R}} t'$ iff

(i) $t \succ_D \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c''\rangle$ (ii) $(\langle P_1, c''\rangle, \langle P_2, c''\rangle) \in \mathcal{R}$

A transition is redundant w.r.t. $\mathcal{R}$ if it is dominated in $\mathcal{R}$ by another transition. Otherwise, it is irredundant.

Irredundant Bisimilarity

Definition

An irredundant bisimulation is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ implies that:

(i) if $\gamma_1 \downarrow_c$ then $\gamma_2 \downarrow_c$,

(ii) if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ is irredundant in $\mathcal{R}$ then $\exists \gamma_2'$ s.t. $\gamma_2 \xrightarrow{\alpha} \gamma_2'$ and $(\gamma_1', \gamma_2') \in \mathcal{R}$.

Define $\gamma_1 \sim_I \gamma_2$ if there exists an irredundant bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Write $P \sim_I Q$ iff $\langle P, \mathit{true} \rangle \sim_I \langle Q, \mathit{true} \rangle$

Outline

Features of the partition refinement algorithm for ccp

- We use barbs.
- Instead of using the function **F** we employ the function **IR** defined as follows: for all partitions $\mathcal{P}$, $\gamma_1 \, \mathbf{IR}(\mathcal{P}) \, \gamma_2$ iff
  - if $\gamma_1 \xrightarrow{\alpha} \gamma_1'$ is irredundant in $\mathcal{P}$, then there exists γ_2' s.t. $\gamma_2 \xrightarrow{\alpha} \gamma_2'$ and $\gamma_1' \, \mathcal{P} \gamma_2'$.

Closure rules

- In the computation of $\mathbf{IR}(\mathcal{P}^n)$, there may also be involved states that are not reachable from the initial states IS.
- We have to change the initialization step of our algorithm, by including in the set $IS^\star$ all the states that are needed to check redundancy. This is done, by using the following closure rules.

$$(\text{is})\frac{\gamma \in IS}{\gamma \in IS^\star} \qquad (\text{rs})\frac{\gamma_1 \in IS^\star \quad \gamma_1 \xrightarrow{\alpha} \gamma_2}{\gamma_2 \in IS^\star}$$

$$(\text{rd})\frac{\gamma \in IS^\star \quad \gamma \xrightarrow{\alpha_1} \gamma_1 \quad \gamma \xrightarrow{\alpha_2} \gamma_2 \quad \gamma \xrightarrow{\alpha_1} \gamma_1 \succ_D \gamma \xrightarrow{\alpha_2} \gamma_3}{\gamma_3 \in IS^\star}$$

Algorithm for ccp

Algorithm 2 `CCP-Partition-Refinement(`IS`)`

Initialization

1. Compute $IS^\star$ with the rules (is), (rs) and (rd),
2. $\mathcal{P}^0 := \{IS^\star_{d_1}\} \ldots \{IS^\star_{d_m}\}$,

Iteration $\mathcal{P}^{n+1} := \mathbf{IR}(\mathcal{P}^n)$

Termination If $\mathcal{P}^n = \mathcal{P}^{n+1}$ then return $\mathcal{P}^n$.

Correctness and termination

Theorem

$\dot{\sim}_{sb} = \dot{\sim}_{l}$

- Since there is a procedure for $\dot{\sim}_{l}$, $\dot{\sim}_{sb}$ can be checked too.
- Any iteration splits blocks and never fuses them.

Theorem

If the set $\texttt{Config}(IS)$ *of all configurations reachable from* IS *is finite, then* $IS^{\star}$ *is finite.*

Proposition

If $IS^{\star}$ *is finite, then the algorithm terminates and the resulting partition coincides with* $\dot{\sim}_{sb}$.

Complexity of the algorithm

- The computation of $IS^{\star}$ according to rules (is), (rs) and (rd) and the decision procedure for $\succ_D$ is needed in the redundancy checks.
- Definition of Domination suggests that deciding whether two given transitions belong to $\succ_D$ may be costly.

Theorem

$\langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle \succ_D \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c''\rangle$ *iff the following conditions hold:* (a) $\alpha \sqsubset \beta$ (b) $c'' = c' \sqcup \beta$

Complexity of the algorithm

- The size of the set $IS^{\star}$ is central to the complexity of the algorithm.
  Depends on of the topology of the transition graph.
- For tree-like topologies, the complexity is quadratic.
- For arbitrary graphs, the complexity may be exponential.

Outline

Computing weak bisimilarity (standard approach)

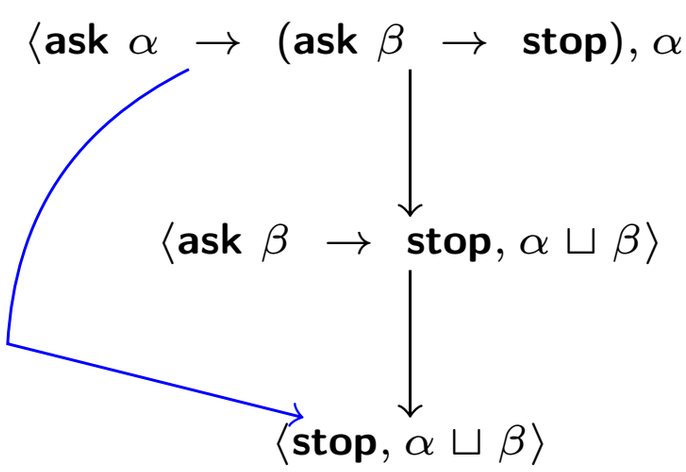
The idea is to start from the graph generated via the operational semantics and then saturate it using the rules described in the table below to produce a new labeled transition relation $\Longrightarrow$.

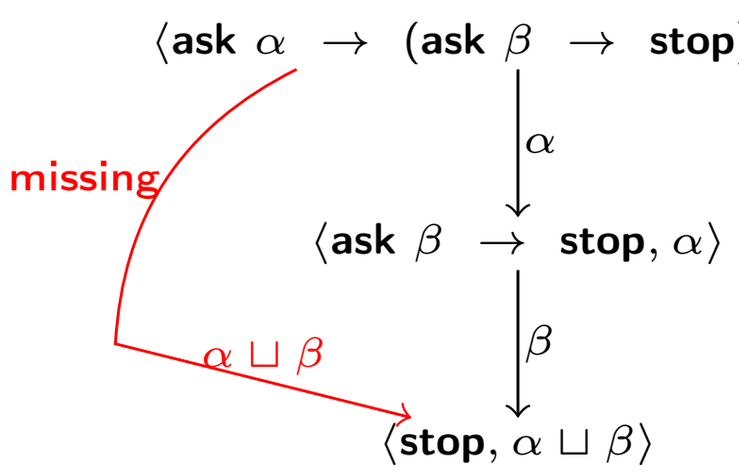
$$\text{MR1}\ \frac{\gamma \xrightarrow{\alpha} \gamma'}{\gamma \overset{\alpha}{\Longrightarrow} \gamma'} \qquad \text{MR2}\ \frac{}{\gamma \overset{true}{\Longrightarrow} \gamma} \qquad \text{MR3}\ \frac{\gamma \overset{true}{\Longrightarrow} \gamma_1 \overset{\alpha}{\Longrightarrow} \gamma_2 \overset{true}{\Longrightarrow} \gamma'}{\gamma \overset{\alpha}{\Longrightarrow} \gamma'}$$

Now the problem whether two states are weakly bisimilar can be reduced to checking whether they are strongly bisimilar wrt $\Longrightarrow$ using the partition refinement.

Incompleteness of the standard method

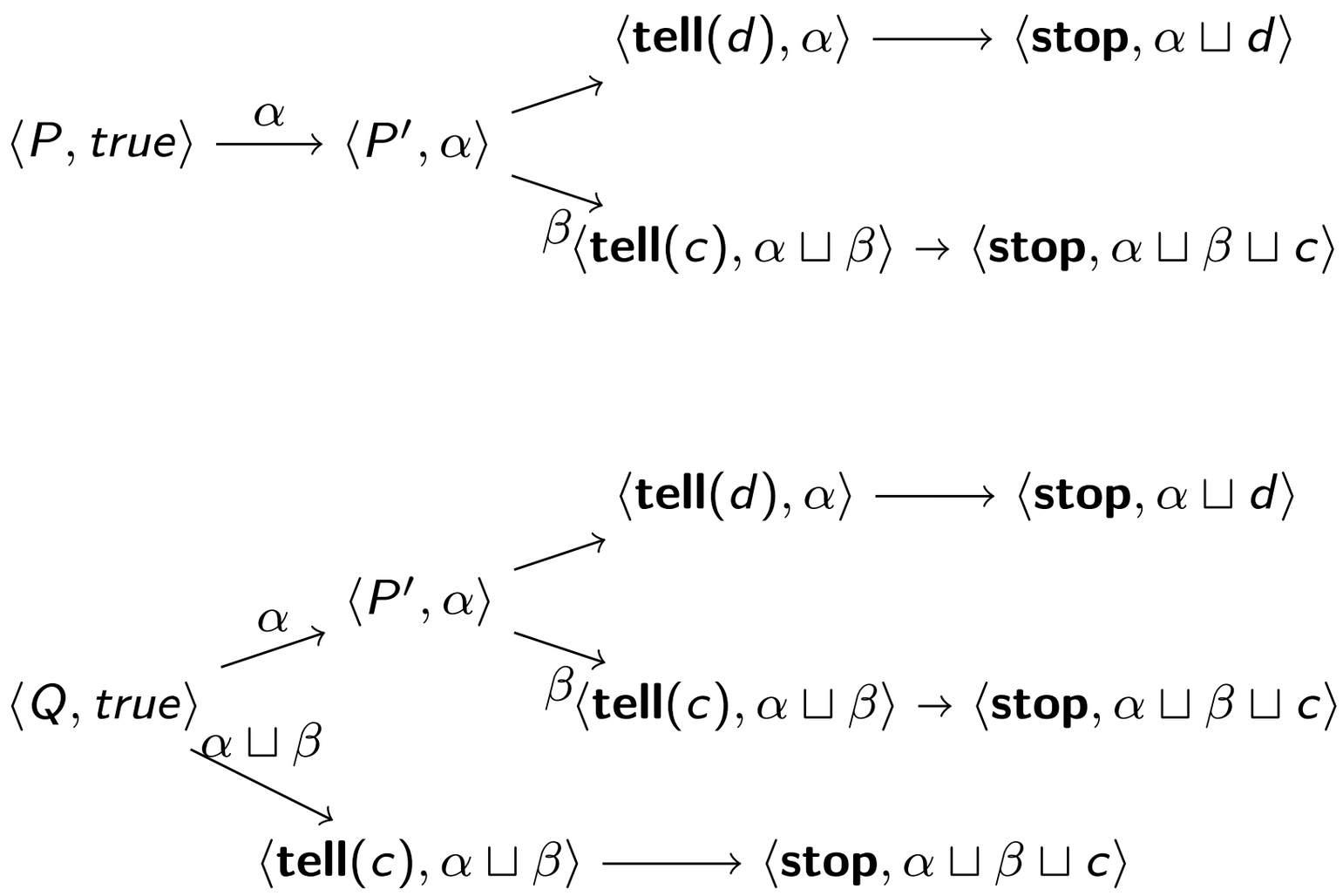
The standard closure does not work for ccp:



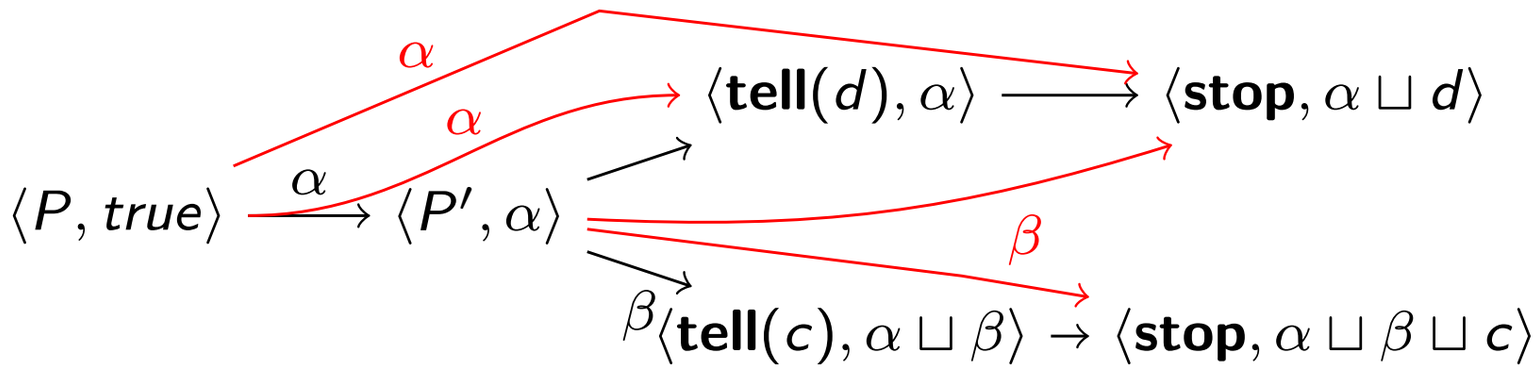


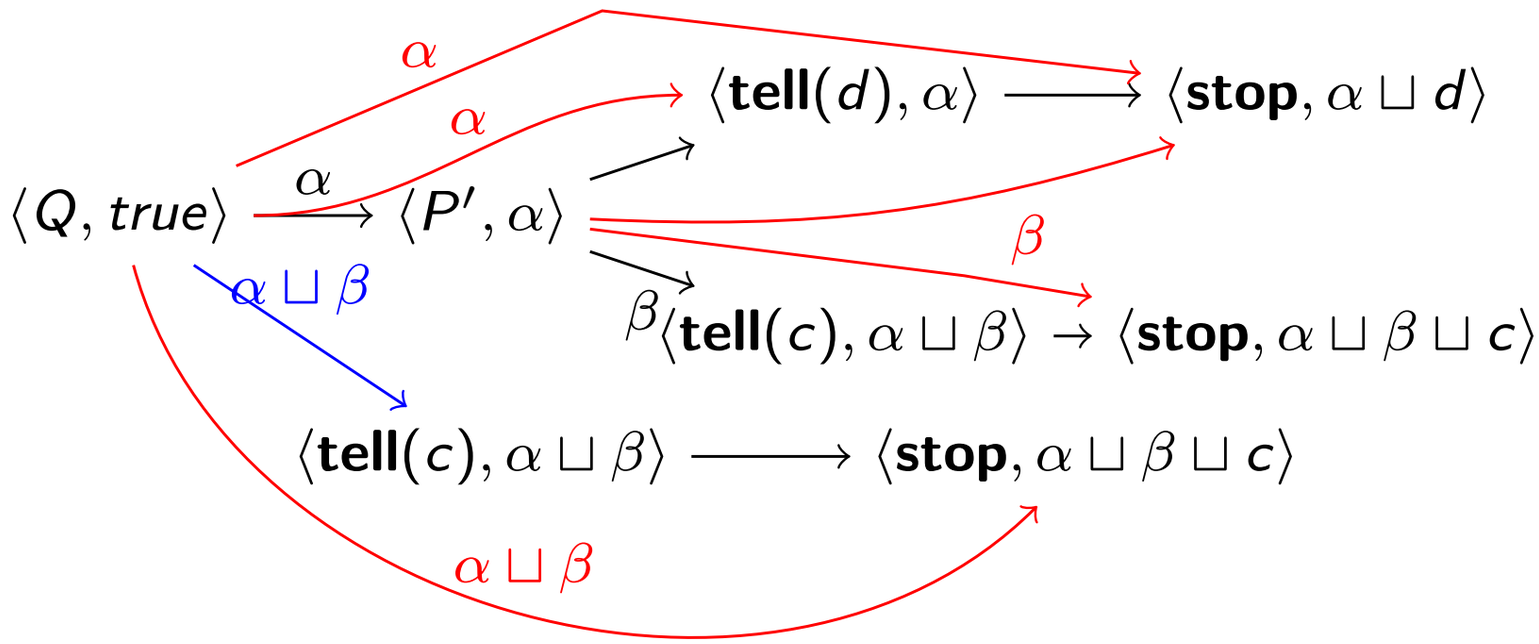
Using the standard closure

Consider the following configurations:



Using the standard closure





Outline

Our Solution

Remark

The fact that the standard closure does not consider the join of the labels is crucial to showing that we need a stronger way of saturating the LTS.

Let us define a new method as follows,

$$\text{R-Tau}\ \frac{}{\gamma \Longrightarrow \gamma} \qquad \text{R-Label}\ \frac{\gamma \xrightarrow{\alpha} \gamma'}{\gamma \overset{\alpha}{\Longrightarrow} \gamma'} \qquad \text{R-Add}\ \frac{\gamma \overset{\alpha}{\Longrightarrow} \gamma' \overset{\beta}{\Longrightarrow} \gamma''}{\gamma \overset{\alpha \sqcup \beta}{\Longrightarrow} \gamma''}$$

Our Solution

Main result

We prove that weak saturated barbed bisimilarity ($\dot{\approx}_{sb}$), observational equivalence, in ccp coincides with the strong version using the transition relation $\Longrightarrow$.

Remark

The fact that the labels are *constraints*, and that $\sqcup$ is commutative and idempotent is crucial to prove that the new LTS is not infinite.

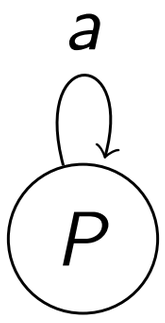
Our approach does not work for CCS

Example

Let us take a very simple recursive CCS process:

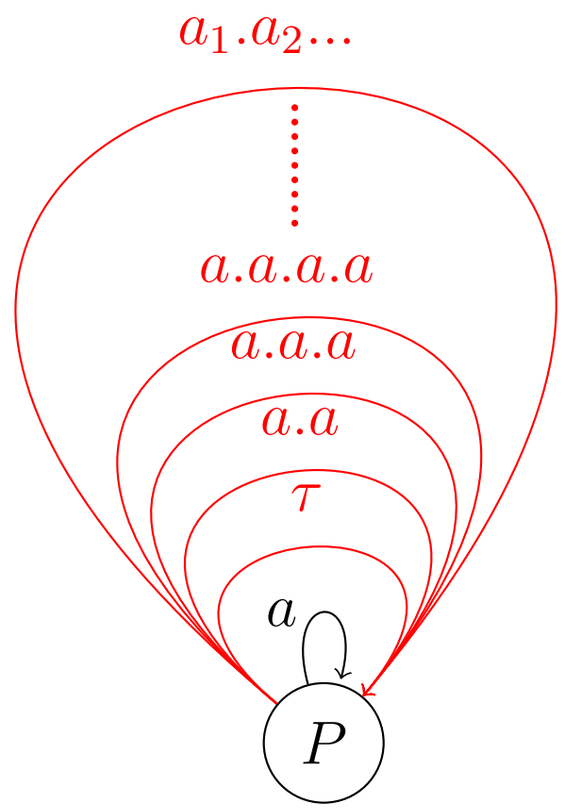
$$P = a.P$$

We can easily see that this process is finitely branching because it just has one transition from the same state P to itself labeled by a.



Our approach does not work for CCS

But if we apply our rules we could end up adding an infinite amount of transitions.



Correspondence results

Theorem

$\dot{\approx}_{sb} = \dot{\sim}_{I}^{\Longrightarrow}$

- Since there is a procedure for $\dot{\sim}_{I}^{\Longrightarrow}$, this way $\dot{\approx}_{sb}$ can be checked too.

Outline

Concluding Remarks

- Transitions are labeled with the minimal information needed from the environment.
- Our notions of bisimilarity provides an alternative co-inductive proof method for ccp.
- Our bisimilarity is coherent w.r.t. barbed congruence and other standard semantics of ccp.
- It makes no fairness assumptions about the transition system.
- The modified partition refinement algorithm verifies the notion of strong bisimilarity for ccp.

Concluding Remarks

- We show that the standard reduction from weak to strong bisimilarity does not work for ccp.
- We then propose a new method for saturating the LTS and we prove that the reduction works.
- The ccp partition refinement algorithm for ccp verifies weak bisimilarity.
- The algorithm has been implemented in C++.

Concluding remarks

Publications from this dissertation

- Papers in Proceedings.
- A. Aristizabal, F. Bonchi, C. Palamidessi, L. Pino, F. Valencia. Deriving Labels and Bisimilarity for Concurrent Constraint Programming. FoSSaCS 2011: 138-152. ©Springer-Verlag. 2011.
- A. Aristizabal, F. Bonchi, L. Pino, F. Valencia. Partition Refinement for Bisimilarity in CCP. SAC 2012: 88-93. ©ACM Press. 2012.
- A. Aristizabal, F. Bonchi, L. Pino, F. Valencia. Reducing Weak to Strong Bisimilarity in CCP. ICE 2012. ©EPTCS.

Concluding remarks

Publications from this dissertation

- Short Papers.
  - Andres A. Aristizabal P. Bisimilarity in Concurrent Constraint Programming. ICLP 2010 (Technical Communications) 236-240 (Short Paper).
  - Andres A. Aristizabal P. Bisimilarity in Concurrent Constraint Programming. Young Researchers Workshop on Concurrency Theory 2010. (Short Paper), 2010.
- Journals (Under preparation).
  - A. Aristizabal, F. Bonchi, C. Palamidessi, L. Pino, F. Valencia. Labeled Bisimilarity for CCP: A Theoretical and Practical Approach.

Future work

- Relate our bisimilarity with other CC-languages.
- Give a characterization of logical equivalence in terms of bisimilarity.
- Take into account the Paige and Tarjan algorithm.

The End

- Thank you very much for your attention!
- Je vous remercie beaucoup de votre attention!
- ¡Muchas gracias por su atención!
- Grazie mille per la vostra attenzione!

Time for questions!

Correspondence results (extended)

The labeled semantics is *sound* and *complete* w.r.t. the unlabeled one.

Lemma (Soundness w.r.t. $\longrightarrow$)

If $\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle$ *then* $\langle P, d \sqcup \alpha\rangle \longrightarrow \langle P', d'\rangle$.

Keys of the proof:

- By induction on (the depth) of the inference of $\langle P, d\rangle \xrightarrow{\alpha} \langle P', d'\rangle$ and a case analysis on the last transition rule used.

Correspondence results (extended)

The labeled semantics is *sound* and *complete* w.r.t. the unlabeled one.

Lemma (Completeness w.r.t. $\longrightarrow$)

If $\langle P, d \sqcup a\rangle \longrightarrow \langle P', d'\rangle$ *then* $\exists \alpha, b$ *s.t.* $\langle P, d\rangle \xrightarrow{\alpha} \langle P', d''\rangle$ *and* $\alpha \sqcup b = a$, $d'' \sqcup b = d'$.

Keys of the proof:

- By induction on (the depth) of the inference of $\langle P, d \sqcup a\rangle \longrightarrow \langle P', d'\rangle$ and a case analysis on the last transition rule used.

Correspondence results (extended)

Our notions are well-behaved w.r.t. barbed congruence.

Lemma (1)

If $\langle P, d\rangle \dot{\sim} \langle Q, e\rangle$, *then* $\forall a \in Con_0$, $\langle P, d \sqcup a\rangle \dot{\sim} \langle Q, e \sqcup a\rangle$.

Keys of the proof:

- Proven by coinduction.
- Using soundness and completeness.

Correspondence results (extended)

Our notions are well-behaved w.r.t. barbed congruence.

Theorem

$\dot{\sim} = \dot{\sim}_{sb}$

Keys of the proof:

- Split the two directions of the proof in two lemmata.
- Proving both lemmata by means of coinduction.
- Using completeness, soundness and lemma (1).

Correspondence results (extended)

Our notions are well-behaved w.r.t. barbed congruence.

Theorem

$\dot{\approx} = \dot{\approx}_{sb}$.

Keys of the proof:

- Following almost the same scheme as for the strong case.

Correspondence results (extended)

A fully-abstract characterization.

Theorem

$P \dot{\approx}_{sb} Q$ *iff* $P \sim_o Q$.

Keys of the proof:

- Split the theorem into two directions.
- Proving both directions by induction.
- Using upward closure of $\dot{\approx}_{sb}$.
- Using lemmata based on fair computations and cofinal chains.

Remark

Our characterization in terms of bisimilarity avoids complicated notions such as fairness and infinite elements.

Correctness and termination (extended)

Corollary

$\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_{I}$

Theorem

$\dot{\sim}_{sb} = \dot{\sim}_{sym}$

Keys of the proof:

- Split the two directions of the proof in two lemmata.
- Proving both lemmata by means of coinduction.
- Using completeness, soundness and monotonicity.

Correctness and termination (extended)

Corollary

$\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_{I}$

Theorem

$\dot{\sim}_{sym} = \dot{\sim}_{I}$

Keys of the proof:

- Split the two directions of the proof in two lemmata.
- Proving both lemmata by means of coinduction.
- Using contradiction.
- Using completeness, soundness and irredundant and symbolic bisimilarity being closed under the addition of constraints.

Correctness and termination (extended)

Corollary

$\dot{\sim}_{sb} = \dot{\sim}_{sym} = \dot{\sim}_{I}$

and transitivity.

- Since there is a procedure for $\dot{\sim}_{I}$, this way $\dot{\sim}_{sb}$ can be checked too.

Correctness and termination (extended)

- Any iteration splits blocks and never fuses them.
- If the set $\texttt{Config}(IS)$ of all configurations reachable from IS is finite, then $IS^\star$ is finite.
- If $IS^\star$ is finite, then the algorithm terminates and the resulting partition coincides with $\dot{\sim}_{sb}$.

Complexity of the algorithm (extended)

- The computation of $IS^{\star}$ according to rules (is), (rs) and (rd) and the decision procedure for $\succ_D$ is needed in the redundancy checks.
- Definition of Domination suggests that deciding whether two given transitions belong to $\succ_D$ may be costly.

Theorem

$\langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle \succ_D \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c''\rangle$ *iff the following conditions hold:* (a) $\alpha \sqsubset \beta$ (b) $c'' = c' \sqcup \beta$

Keys of the proof:

- Dividing the theorem in two directions.
- Prove each direction by direct proof.

Complexity of the algorithm (extended)

- The size of the set $IS^{\star}$ is central to the complexity of the algorithm. Depends on of the topology of the transition graph.
- For tree-like topologies, the complexity is quadratic.
- For arbitrary graphs, the complexity may be exponential.

Symbolic bisimilarity

Definition (Symbolic Bisimilarity)

A symbolic bisimulation is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, c \rangle$ and $\gamma_2 = \langle Q, d \rangle$ it implies that:

(i) if $\gamma_1 \downarrow_e$ then $\gamma_2 \downarrow_e$,

(ii) if $\langle P, c \rangle \xrightarrow{\alpha} \langle P', c' \rangle$ then there exists a transition $t = \langle Q, d \rangle \xrightarrow{\beta} \langle Q', d'' \rangle$ s.t. $t \vdash_D \langle Q, d \rangle \xrightarrow{\alpha} \langle Q', d' \rangle$ and $(\langle P', c' \rangle, \langle Q', d' \rangle) \in \mathcal{R}$

We say that γ_1 and γ_2 are symbolic bisimilar ($\gamma_1 \dot{\sim}_{sym} \gamma_2$) if there exists a symbolic bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Correspondence results (extended)

Lemma (Soundness w.r.t. $\Longrightarrow$)

If $\langle P, d\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', d'\rangle$ *then* $\langle P, d \sqcup \alpha\rangle \Longrightarrow \langle P', d'\rangle$.

Keys of the proof:

- By induction on (the depth) of the inference of $\langle P, d\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', d'\rangle$ and a case analysis on the last transition rule used.

Correspondence results (extended)

Lemma (Completeness w.r.t. $\Longrightarrow$)

If $\langle P, d \sqcup a\rangle \Longrightarrow \langle P', d'\rangle$ *then* $\exists \alpha, b$ *s.t.* $\langle P, d\rangle \stackrel{\alpha}{\Longrightarrow} \langle P', d''\rangle$ *and* $\alpha \sqcup b = a$, $d'' \sqcup b = d'$.

Keys of the proof:

- By induction on (the depth) of the inference of $\langle P, d \sqcup a\rangle \Longrightarrow \langle P', d'\rangle$ and a case analysis on the last transition rule used.

Correspondence results (extended)

Corollary

$$\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$$

Corollary

$$\dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$$

Since $\longrightarrow$ is sound and complete and $\dot{\sim}_{sym} = \dot{\sim}_{I}$ over $\longrightarrow$ and we have just proven that $\Longrightarrow$ is sound and complete, then $\dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$.

Correspondence results (extended)

Corollary

$$\dot{\approx}_{sb} = \dot{\simeq}^{\Longrightarrow}_{sym} = \dot{\simeq}^{\Longrightarrow}_{I}$$

Theorem

$$\dot{\simeq}^{\Longrightarrow}_{sym} = \dot{\approx}_{sb}$$

Keys of the proof:

- Split the two directions of the proof in two lemmata.
- Proving both lemmata by means of coinduction.
- Using completeness, soundness, the fact that $\longrightarrow^* = \Longrightarrow$ and $\Downarrow = \not\downarrow$, where $\not\downarrow$ express the barbs obtained by the relation $\Longrightarrow$.

Correspondence results (extended)

Corollary

$$\dot{\approx}_{sb} = \dot{\sim}^{\Longrightarrow}_{sym} = \dot{\sim}^{\Longrightarrow}_{I}$$

and transitivity.

- Since there is a procedure for $\dot{\sim}^{\Longrightarrow}_{I}$, this way $\dot{\approx}_{sb}$ can be checked too.

Rule R4

Intuitively, $\exists_x^e P$ behaves like P, except that the variable x possibly present in P must be considered local, and that the information present in e has to be taken into account. It is convenient to distinguish between the *external* and the *internal* points of view. From the internal point of view, the variable x, possibly occurring in the global store d, is hidden. This corresponds to the usual scoping rules: the x in d is *global*, hence "covered" by the local x. Therefore, P has no access to the information on x in d, and this is achieved by filtering d with $\exists_x$. Furthermore, P can use the information (which may also concern the local x) that has been produced locally and accumulated in e. In conclusion, if the visible store at the external level is d, then the store that is visible internally by P is $e \sqcup \exists_x d$. Now, if P is able to make a step, thus reducing to P' and transforming the local store into e', what we see from the external point of view is that the process is transformed into $\exists_x^{e'} P'$, and that the information $\exists_x e$ present in the global store is transformed into $\exists_x e'$.

$IS^\star$ is finite

Theorem

If $\texttt{Config}(IS)$ *is finite, then* $IS^\star$ *is finite.*

Proof.

As a first step, we observe that a configuration $\gamma \in IS^\star$ only if $\gamma = \langle P, d \sqcup e\rangle$ and $\langle P, d\rangle \in \texttt{Config}(IS)$. Then we prove that there are only finitely many such constraints e. Let $\texttt{Label}(IS)$ be the set of all labels in the LTS of IS generated by the labeled semantics. This set is finite (since $\texttt{Config}(IS)$ is finite), and its downward closure $\downarrow\texttt{Label}(IS) = \{a \mid \exists b \in \texttt{Label}(IS) \text{ with } a \sqsubseteq b\}$ is also finite (since $\sqsubseteq$ is well-founded). The set of all e s.t. $\langle P, d \sqcup e\rangle \in IS^\star$ (with $\langle P, d\rangle \in \texttt{Config}(IS)$) is a subset of $\downarrow\texttt{Label}(IS)$ and thus it is finite. Indeed, observe that if $\langle P, d \sqcup e\rangle \xrightarrow{c_1}$, then $\langle P, d\rangle \xrightarrow{c_2}$ with $c_1 \sqsubseteq c_2$. Therefore $\texttt{Label}(IS^\star) \subseteq \downarrow\texttt{Label}(IS)$. Moreover, if $\langle P, d \sqcup e\rangle$ is added to $IS^\star$ by the rule (rd) then, by definition of $\succ_D$ $e \sqsubseteq \beta$ for β being a label in $\texttt{Label}(IS^\star)$ (i.e., in $\downarrow\texttt{Label}(IS)$). □

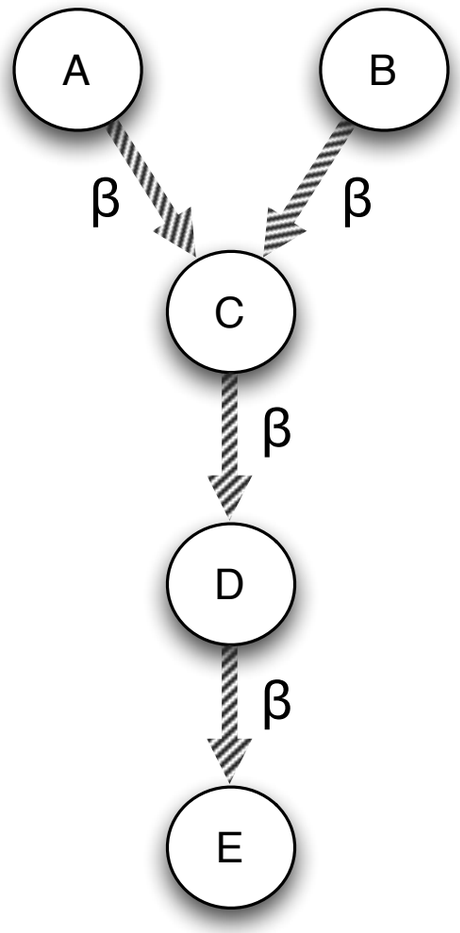
RULE LR4

The information from the environment α needs to be added to the global store d, hence the occurrences of x in both d and α must be identified. Notice that dropping the existential quantification of x in d in the antecedent transition does identify the occurrences of x in d with those in α but also with those in the local store e thus possibly generating variable clashes. It solves the above-mentioned issues by using in the antecedent derivation a fresh variable z that acts as a substitute for the free occurrences of x in P and its local store e. (Recall that $T[z/x]$ represents T with x replaced with z). This way we identify with z the free occurrences of x in P and e and avoid clashes with those in α and d.

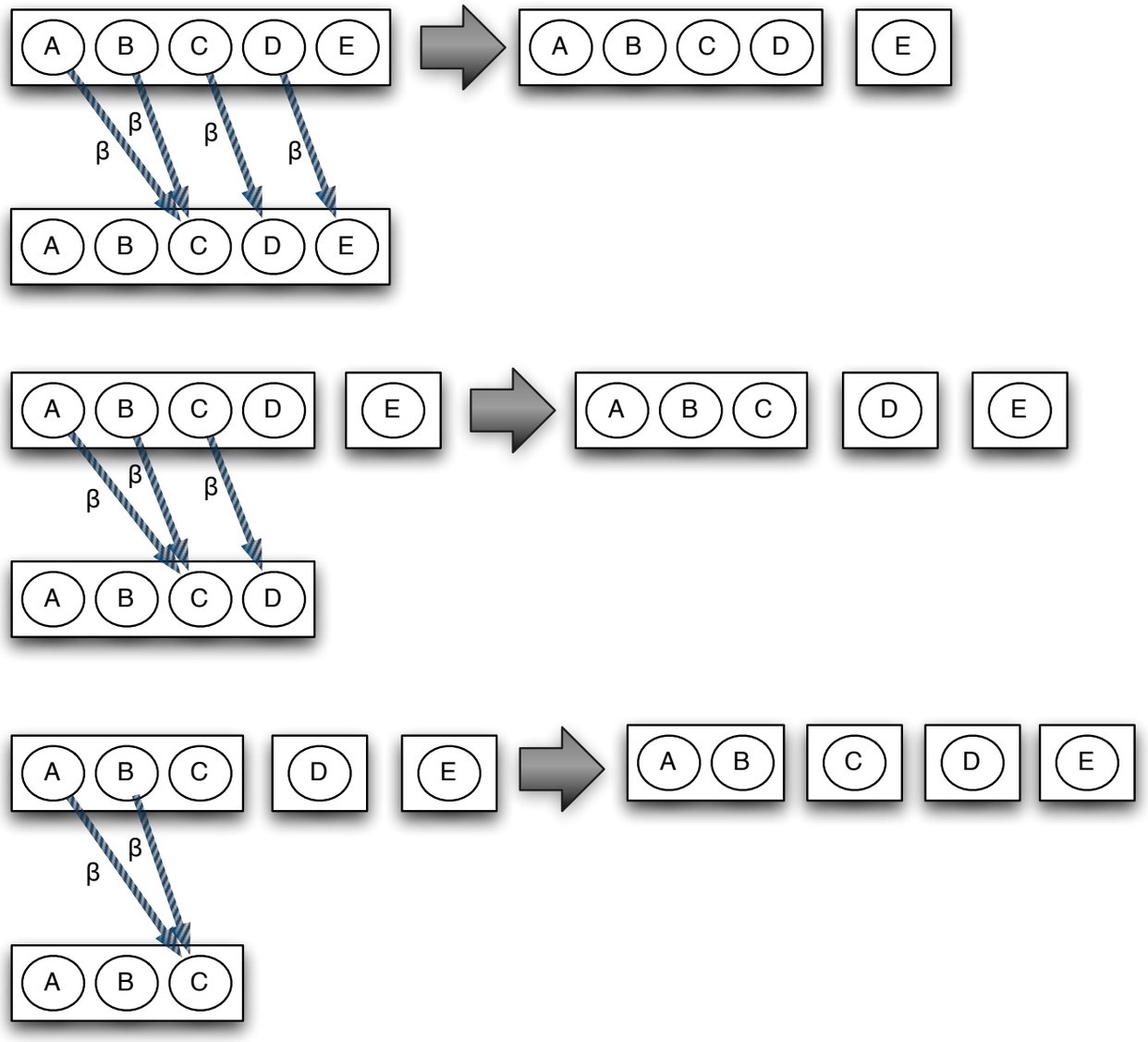
α as minimal information

Using α as the minimal information from the environment instead of any information which allows the configuration to evolve is crucial for the well-behavior of the labeled bisimilarity notion for ccp. With this α the bisimilarity notion does not over-discriminate.

Standard Partition Refinement: Example



A Labelled Transition System



Splitting the Blocks

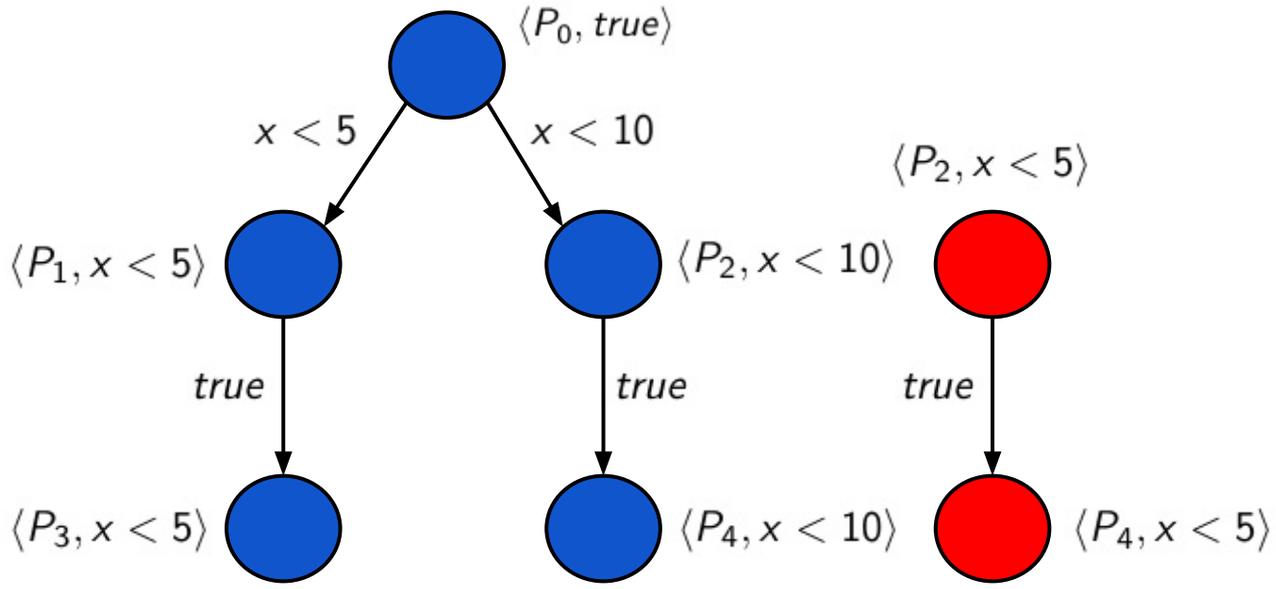
Redundant Transitions

We first define a *derivation relation* $\vdash_D$ amongst the transitions of ccp processes: $\gamma \xrightarrow{\alpha_1} \gamma_1 \vdash_D \gamma \xrightarrow{\alpha_2} \gamma_2$ which intuitively means that the latter transition is a logical consequence of the former. Then we introduce the notion of *domination* ($\succ_D$), which means that if there exist two transitions $\gamma \xrightarrow{\alpha_1} \gamma_1$ and $\gamma \xrightarrow{\alpha_2} \gamma_2$, $\gamma \xrightarrow{\alpha_1} \gamma_1 \succ_D \gamma \xrightarrow{\alpha_2} \gamma_2'$ if and only if $\gamma \xrightarrow{\alpha_1} \gamma_1 \vdash_D \gamma \xrightarrow{\alpha_2} \gamma_2'$ and $\alpha_1 \neq \alpha_2$. Finally we give the definition of a *redundant transition*. Intuitively $\gamma \xrightarrow{\alpha_2} \gamma_2$ is redundant if $\gamma \xrightarrow{\alpha_1} \gamma_1$ dominates it, that is $\gamma \xrightarrow{\alpha_1} \gamma_1 \succ_D \gamma \xrightarrow{\alpha_2} \gamma_2'$ and $\gamma_2 \dot{\sim}_{sb} \gamma_2'$.I

Example: Redundant Transition

$\langle P_0, true\rangle \xrightarrow{x<10} \langle P_2, x<10\rangle \prec_{\mathcal{R}_2} \langle P_0, true\rangle \xrightarrow{x<5} \langle P_1, x<5\rangle$

$(i)\,\langle P_0, true\rangle \xrightarrow{x<10} \langle P_2, x<10\rangle \succ_D \langle P_0, true\rangle \xrightarrow{x<5} \langle P_2, x<5\rangle\,(ii)\,(\langle P_2, x<5\rangle, \langle P_1, x<5\rangle) \in \mathcal{R}_2$



$\mathcal{R}_1 = \{(\langle P_0, true\rangle, \langle P_0, true\rangle)\}$

$\mathcal{R}_2 = \{(\langle P_1, x<5\rangle, \langle P_1, x<5\rangle), (\langle P_2, x<5\rangle, \langle P_2, x<5\rangle), (\langle P_1, x<5\rangle, \langle P_2, x<5\rangle), (\langle P_2, x<5\rangle, \langle P_1, x<5\rangle)\}$

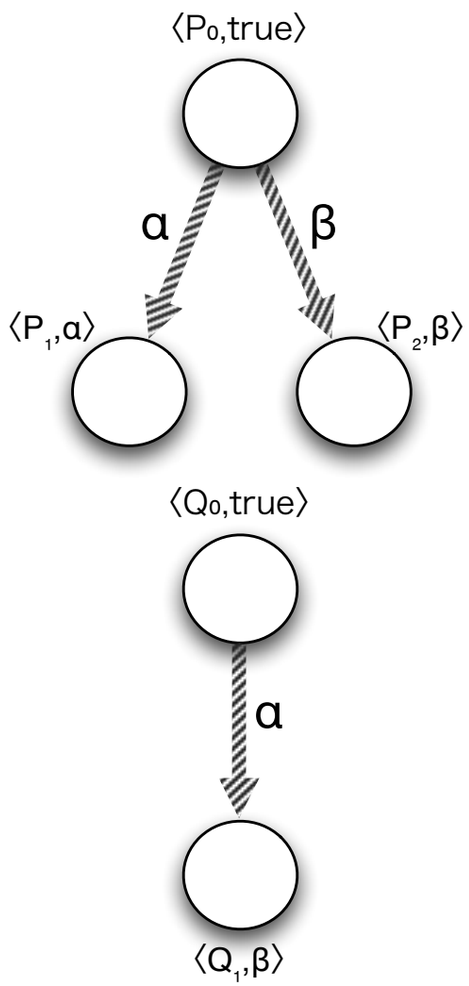
$\mathcal{R}_3 = \{(\langle P_3, x<5\rangle, \langle P_3, x<5\rangle), (\langle P_4, x<5\rangle, \langle P_4, x<5\rangle), (\langle P_3, x<5\rangle, \langle P_4, x<5\rangle), (\langle P_4, x<5\rangle, \langle P_3, x<5\rangle)\}$

$\mathcal{R}_4 = \{(\langle P_2, x<10\rangle, \langle P_2, x<10\rangle)\}$

$\mathcal{R}_5 = \{(\langle P_4, x<10\rangle, \langle P_4, x<10\rangle)\}$

Algorithm for ccp: Example

$\alpha \sqsubset \beta$



Two Labelled Transition Systems

$$\langle P_0, true\rangle \xrightarrow{\alpha} \langle P_1, \alpha\rangle \succ_D \langle P_0, true\rangle \xrightarrow{\beta} \langle P_1, \beta\rangle$$

$$\mathcal{P}^0_{=} \quad \boxed{\langle P_0, \mathtt{true}\rangle, \langle Q_0, \mathtt{true}\rangle} \quad \boxed{\langle P_1, \beta\rangle, \langle P_2, \beta\rangle} \quad \boxed{\langle P_1, \alpha\rangle, \langle Q_1, \alpha\rangle}$$

$$\mathbf{IR}: \quad \begin{array}{l} \boxed{\langle P_0, \mathtt{true}\rangle, \langle Q_0, \mathtt{true}\rangle} \\ \boxed{\langle P_1, \beta\rangle, \langle P_2, \beta\rangle} \\ \boxed{\langle P_1, \alpha\rangle, \langle Q_1, \alpha\rangle} \end{array} \quad \mapsto \quad \begin{array}{l} \boxed{\langle P_0, \mathtt{true}\rangle, \langle Q_0, \mathtt{true}\rangle} \\ \boxed{\langle P_1, \beta\rangle, \langle P_2, \beta\rangle} \\ \boxed{\langle P_1, \alpha\rangle, \langle Q_1, \alpha\rangle} \end{array}$$

$$\langle P_1, \beta\rangle \mathcal{R} \langle P_2, \beta\rangle$$

$$\langle P_0, \mathtt{true}\rangle \xrightarrow{\alpha} \langle P_1, \alpha\rangle \succ_{\boxed{\langle P_1, \beta\rangle, \langle P_2, \beta\rangle}} \langle P_0, \mathtt{true}\rangle \xrightarrow{\beta} \langle P_2, \beta\rangle$$

Procedure Rules

$$\text{R5}\quad \frac{\langle P[\vec{z}/\vec{x}], d\rangle \longrightarrow \gamma'}{\langle p(\vec{z}), d\rangle \longrightarrow \gamma'}$$

where $p(\vec{x}) \stackrel{\text{def}}{=} P$ is a process definition in D

$$\text{LR5}\quad \frac{\langle P[\vec{z}/\vec{x}], d\rangle \xrightarrow{\alpha} \gamma'}{\langle p(\vec{z}), d\rangle \xrightarrow{\alpha} \gamma'}$$

where $p(\vec{x}) \stackrel{\text{def}}{=} P$ is a process definition in D

Bisimilarity first steps

- Automata Classical Theory: Different automata regarding the same behavior. Classification of automata according to its behavior.
- Little attention was paid into the way in which to automata may interact. In the sense that an action by one entails a complementary action by another.
- Based on the concept of mutual simulation. A new notion of how two LTS behave in realation which each other. A behavioral equivalence that distinguishes two LTS even if they are language-equivalent.
- Bisimulation and bisimilarity are coinductive notions, and as such are intimately related to fixed points, in particular greatest fixed points.

Importance of Bisimilarity

- Bisimilarity is accepted as the finest behavioural equivalence one would like to impose on processes.
- The bisimulation proof method is exploited to prove equalities among processes.
- This occurs even when bisimilarity is not the behavioural equivalence chosen for the processes. For instance, one may be interested in trace equivalence and yet use the bisimulation proof method since bisimilarity implies trace equivalence.
- The efficiency of the algorithms for bisimilarity checking and the compositionality properties of bisimilarity are exploited to minimise the state-space of processes.
- Bisimilarity, and variants of it such as similarity, are used to abstract from certain details of the systems of interest. For instance, we may want to prove behavioural properties of a server that do not depend on the data that the server manipulates. Further, abstracting from the data may turn an infinite-state server into a finite one.

Importance of Bisimilarity

- The checks are local because we only look at the immediate transitions that emanate from the states. An example of a behavioural equality that is non-local is trace equivalence (two states are trace equivalent if they can perform the same sequences of transitions).
- It is non-local because computing a sequence of transitions starting from a state S may require examining other states, different from S.
- There is no hierarchy on the pairs of a bisimulation in that no temporal order on the checks is required: all pairs are on a par. As a consequence, bisimilarity can be effectively used to reason about infinite or circular objects. This is in sharp contrast with inductive techniques, that require a hierarchy, and that therefore are best suited for reasoning about finite objects.

Importance of Bisimilarity in ccp

- Elegantly captures the notion of process equivalence.
- Helps to veirfy if two ccp programs are behavioral equivalent or not in an easy co-inductive way.
- The strong ties between CCP and logic may provide with a novel characterization of logic equivalence in terms of bisimilarity.

Importance of program equivalence in ccp

- To better understand the equivalence or difference in the behavior of a process or program.
- We get the notion for comparing program specifications.
- Subject to notion of observability.
- Enjoy robust mathematical properties.
- Appropriate proof techniques.
- To see if a simpler process or program can behave in the same way as a much more complicated one.
- To verify if two programs which seem almost the same are indeed equivalent or not.
- In practice
  - To see if implementations fulfill their specification.
  - To understand which specification it is better suited for an implementation.

Differences between strong and weak bisimilarity

- In strong equivalences, all the transitions performed by a system are deemed observable.
- Strong equivalences are usually much easier to be checked.
- Strong bisimilarity is a good candidate for "indistinguishability".
  - refines trace equivalence from automata theory.
  - takes into account branching structure.
  - congruence relation.
  - elegant proof techniques.
- But can sometimes be too strong.
  - uniform definition w.r.t. all transitions, including τ transitions
  - . . . but the latter are supposed to be unobservable.

Differences between strong and weak bisimilarity

- In weak equivalences, instead, internal transitions (usually denoted by τ) are unobservable.
- Weak equivalences are more abstract (and thus closer to the intuitive notion of behaviour).
- Weak bisimilarity "assumes" progress.
- Weak bisimilarity is behavioural equivalence is obtained from the strong case, therefore a strong is a weak bisimilarity.

Symbolic Bisimilarity

Definition (Symbolic Bisimilarity)

A symbolic bisimulation is a symmetric relation $\mathcal{R}$ on configurations s.t. whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, c \rangle$ and $\gamma_2 = \langle Q, d \rangle$ implies that:

(i) if $\gamma_1 \downarrow_e$ then $\gamma_2 \downarrow_e$,

(ii) if $\langle P, c \rangle \xrightarrow{\alpha} \langle P', c' \rangle$ then there exists a transition $t = \langle Q, d \rangle \xrightarrow{\beta} \langle Q', d'' \rangle$ s.t. $t \vdash_D \langle Q, d \rangle \xrightarrow{\alpha} \langle Q', d' \rangle$ and $(\langle P', c' \rangle, \langle Q', d' \rangle) \in \mathcal{R}$

We say that γ_1 and γ_2 are symbolic bisimilar ($\gamma_1 \dot{\sim}_{sym} \gamma_2$) if there exists a symbolic bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$.

Barbed Bisimilarity

Definition

A barbed bisimulation is a symmetric relation $\mathcal{R}$ on configurations such that whenever $(\gamma_1, \gamma_2) \in \mathcal{R}$ with $\gamma_1 = \langle P, d \rangle$ and $\gamma_2 = \langle Q, e \rangle$ implies that:

(i) if $\gamma_1 \downarrow_c$ then $\gamma_2 \downarrow_c$ and

(ii) if $\gamma_1 \rightarrow \gamma_1'$ then $\exists \gamma_2'$ s.t. $\gamma_2 \rightarrow \gamma_2'$ and $(\gamma_1, \gamma_2) \in \mathcal{R}$.

- Say that $\gamma_1 \dot{\sim}_b \gamma_2$ if there is a barbed bisimulation $\mathcal{R}$ s.t. $(\gamma_1, \gamma_2) \in \mathcal{R}$.
- Say that $P \dot{\sim}_b Q$ iff $\langle P, \mathit{true} \rangle \dot{\sim}_b \langle Q, \mathit{true} \rangle$.

Congruence

One can verify that $\dot{\sim}_b$ is an equivalence. However, it is not a *congruence*; i.e., it is not preserved under arbitrary contexts. A context C is a term with a hole $[-]$ s.t., replacing it with a process P yields a process term $C[P]$. E.g., $C = \mathbf{tell}(c) \parallel [-]$ and $C[\mathbf{tell}(d)] = \mathbf{tell}(c) \parallel \mathbf{tell}(d)$

Example: Not a congruence

- $P = \mathbf{ask}\ (b)\ \rightarrow\ \mathbf{tell}(d)$ and $Q = \mathbf{ask}\ (c)\ \rightarrow\ \mathbf{tell}(d)$ with $a, b, c, d \neq \mathit{true}, b \sqsubseteq a$ and $c \not\sqsubseteq a$.
- We have $\langle P, \mathit{true}\rangle \dot{\sim}_b \langle Q, \mathit{true}\rangle$
- $\langle \mathbf{tell}(a) \parallel P, \mathit{true}\rangle \dot{\not\sim}_b \langle \mathbf{tell}(a) \parallel Q, \mathit{true}\rangle$.

Reducing cost for domination

Theorem

$\langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle \succ_D \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c''\rangle$ *iff the following conditions hold:*

(a) $\alpha \sqsubset \beta$ *(b)* $c'' = c' \sqcup \beta$

Proof.

(⇒) We assume $\langle P, c\rangle \xrightarrow{\alpha} \langle P_1, c'\rangle \succ_D \langle P, c\rangle \xrightarrow{\beta} \langle P_1, c''\rangle$ *namely (i), (ii) and (iii) in Definition of Domination hold. Take* $e_2 = e \sqcup \alpha$ *therefore we have* $\alpha \sqcup e_2 = \alpha \sqcup e \sqcup \alpha = \alpha \sqcup e$ *that, by condition (ii) is equal to* β*. Since* $\alpha \neq \beta$ *(iii), then* $\alpha \sqsubset \beta$*, i.e., condition (a) holds. Notice that* $\alpha \sqsubseteq c'$ *since labels are added to the stores when performing transitions. By condition (ii), we have that* $c'' = c' \sqcup e = c' \sqcup \alpha \sqcup e$ *that, by condition (i), is equal to* $c' \sqcup \beta$*. Thus, (b) holds. (⇐) Conversely, assume that (a) and (b) hold. Since* $\alpha \sqsubset \beta$*, there exists an* e_2 *s.t.* $\beta = \alpha \sqcup e_2$ *and* $\alpha \neq \beta$ *(the latter is condition (iii) in Definition of domintation). Now to prove (i), (ii) we take* $e = e_2 \sqcup \alpha$*. Since* $\beta = \alpha \sqcup e_2$*, then* $\beta = \alpha \sqcup e_2 = \alpha \sqcup e_2 \sqcup \alpha = \alpha \sqcup e$*, i.e., (i) holds. By (b),* $c'' = c' \sqcup \beta = c' \sqcup \alpha \sqcup e_2 = c' \sqcup e$*, i.e., (ii) also holds.* □

The Paige and Tarjan Algorithm

- An algorithm that can solve problems efficiently using a repeated partition refinement strategy.
- They propose an algorithm running in $O(m \log n)$ time and $O(m + n)$ space.
- Their algorithm combines Hopcroft's "process the smaller half" strategy with a new variant of partition refinement. Hopcroft used the "process the smaller half" idea to solve a deceptively similar problem. A solution to this problem can be used to minimize the number of states of a deterministic finite automaton.
- As Kanellakis and Smolka noted, the relational coarsest partition problem is sufficiently different from the functional problem that a nontrivial generalization of Hopcroft's algorithm is needed to solve it.
- This algorithm uses a primitive refinement operation that generalizes the one used in Hopcroft's algorithm.
- The algorithm exploits several properties of split and consequences of stability.

The Standard Reduction method: Not Complete for ccp

Definition

We say that a relation $\overset{\beta}{\rightsquigarrow} \subseteq Conf \times Con_0 \times Conf$ is complete iff whenever $(\langle P, c \sqcup a \rangle, true, \langle P', c' \rangle) \in \overset{\beta}{\rightsquigarrow}$ then there exist $\alpha, b \in Con_0$ s.t. $\langle P, c \rangle \overset{\alpha}{\rightsquigarrow} \langle P', c'' \rangle$ where $\alpha \sqcup b = a$ and $c'' \sqcup b = c'$.

The Standard Reduction method: Not Complete for ccp

We will show a counter-example where the completeness for $\Longrightarrow$ does not hold. Let $P = \textbf{ask}\ \alpha\ \rightarrow\ (\textbf{ask}\ \beta\ \rightarrow\ \textbf{stop})$ and $d = \alpha \sqcup \beta$. Now consider the transition $\langle P, d\rangle \Longrightarrow \langle \textbf{stop}, d\rangle$ and let us apply the completeness lemma, we can take $c = \mathit{true}$ and $a = \alpha \sqcup \beta$, therefore by completeness there must exist b and λ s.t. $\langle P, \mathit{true}\rangle \stackrel{\lambda}{\Longrightarrow} \langle \textbf{stop}, c''\rangle$ where $\lambda \sqcup b = \alpha \sqcup \beta$ and $c'' \sqcup b = d$. However, notice that the only transition possible is $\langle P, \mathit{true}\rangle \stackrel{\alpha}{\Longrightarrow} \langle \textbf{ask}\ \beta\ \rightarrow\ \textbf{stop}, \alpha\rangle$, hence completeness does not hold since there is no transition from $\langle P, \mathit{true}\rangle$ to $\langle \textbf{stop}, c''\rangle$ for some c''.

Asynchronous Bisimilarity

Definition (Asynchronous Bisimilarity)

A symmetric relation $\mathcal{R}$ is an asynchronous bisimulation iff, whenever $p\mathcal{R}q$,

- if $p \xrightarrow{\mu} p'$ where μ is not an input action and $bn(\mu)$ is fresh, then $\exists q'$ such that $q \xrightarrow{\mu} q'$ and $p'\mathcal{R}q'$,
- if $p \xrightarrow{a(b)} p'$, then $\exists q'$ such that either $q \xrightarrow{a(b)} q'$ and $p'\mathcal{R}q'$, or $q \xrightarrow{\tau} q'$ and $p'\mathcal{R}(q'|\bar{a}b)$.

We say that p and q are asynchronous bisimilar (written $p \sim^a q$) if and only if there is an asynchronous bisimulation relating them.

Asynchronous Bisimilarity

For instance, the symmetric closure of the following relation is an asynchronous bisimulation.

$\mathcal{R} = \{(\tau.\nu y.ya + a(b).ab, \tau.0), (\nu y.ya, 0)\} \cup \{(ax, \nu y.ya|ax) | x \in N\}$.

Constraint System (implementation)

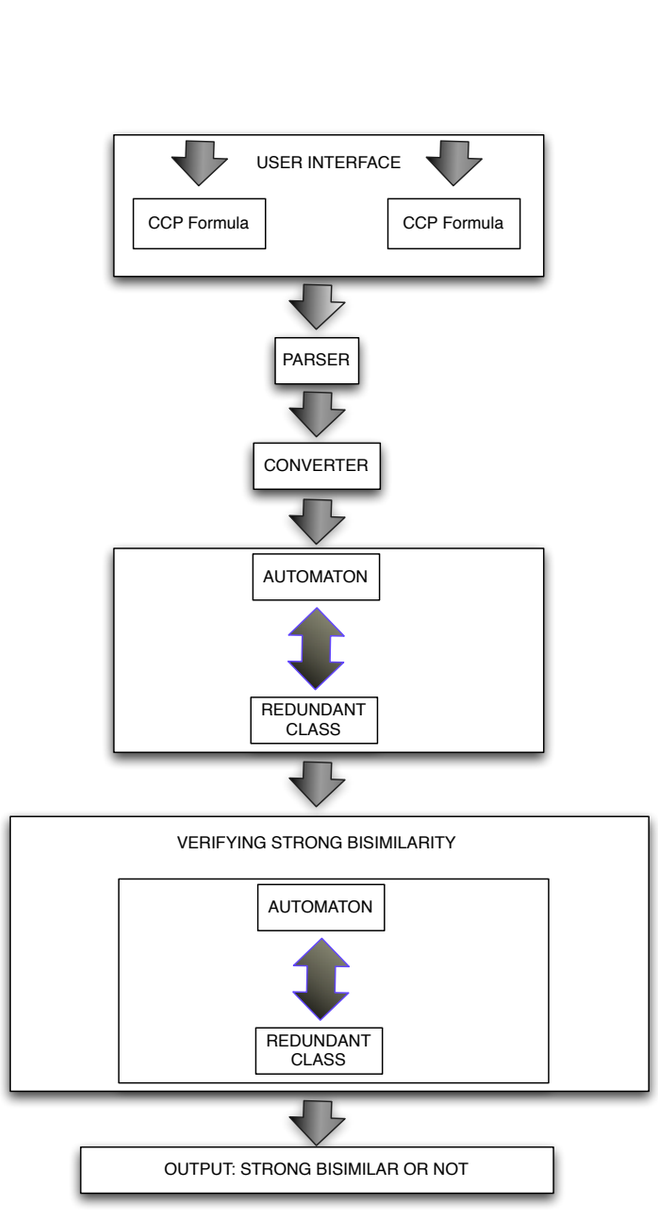
Constraint system

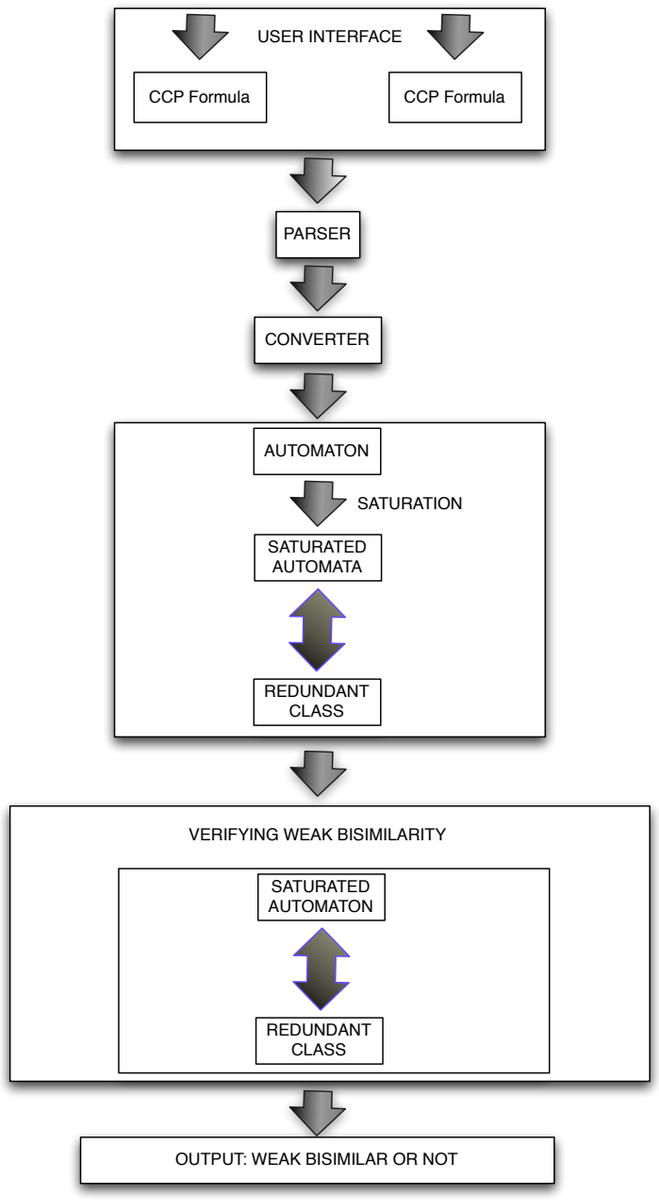
The cs we are able to handle under this implementation involves the following simple constraints of the form *var oper num*, where *var* $\in$ *variables*, represents a set of variables in the problem; *oper* $\in \{<, >, \leq, \geq, =\}$, represents a set of operators; and *num* $\in [0..99]$, represents the domain of the variables.

Programming language

- As said in [Jeong et al 98'] and [Fernandez 89'], data structures are required to represent states, transitions, classes, splitters, etc.
- The most adequate programming language is an object oriented.
- The use of classes is important to model ccp, since we can abstract each component of the cs into a class.
- We chose C++.
  - It is an efficient compiler to native code.
  - Exposes low-level system facilities rather than running in a virtual machine.
  - Supports pointers, references, and pass-by-value.
  - Has an explicit memory management.
  - Provides cross-platform access to many different features available in platform-specific libraries.

Implementation layout





Experiments

- We use families of generated automata resembling ccp configurations in order to analyze the behavior in time of the ccp partition refinement algorithm.
- We analyze transitions, nodes, percentage of transitions with the same label, percentage of configurations satisfying the same barb and number of branches.
- This analysis help us explain the relevance and the insights of the algorithm for checking bisimilarity.
- The machine used to carry out the experiments had a processor of 2.4 GHz Intel Core 2 Duo with ram memory of 4GB 1067 MHz DDR3.

Some results: table

Processes	Nodes	Transitions	Time
Process 10	34	32	380 ms
Process 11	16	14	58 ms
Process 12	28	26	222 ms
Process 13	30	28	281 ms
Process 14	32	30	320 ms
Process 15	50	48	970 ms
Process 16	682	680	5960420 ms
Process 17	170	168	66947 ms
Process 18	426	424	2570960 ms
Process 19	218	216	522674 ms
Process 20	274	272	896211 ms

Table: Time vs. Transitions/Nodes

Some results: graph 1

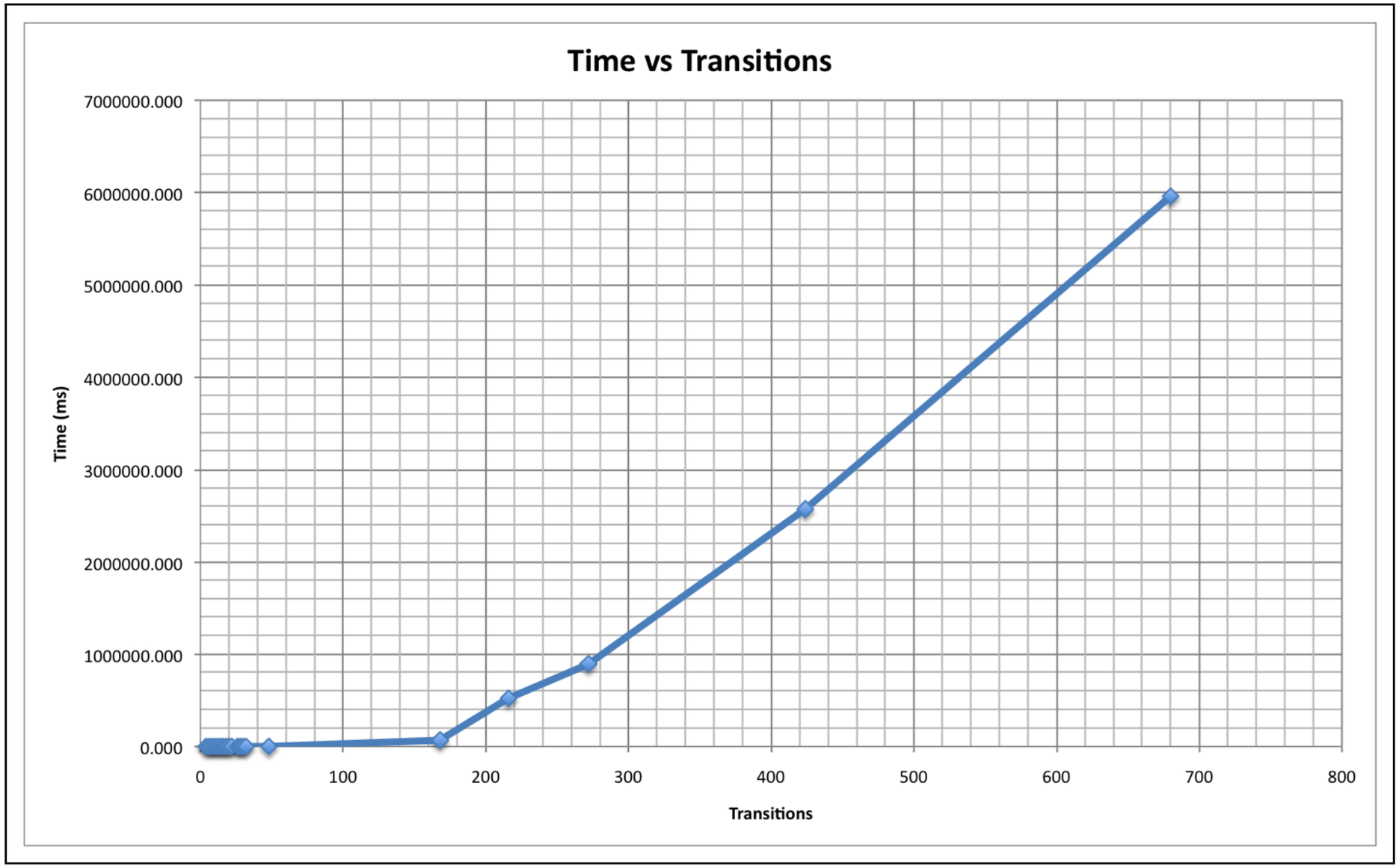


Figure: Time vs. Transitions

Some results: graph 3

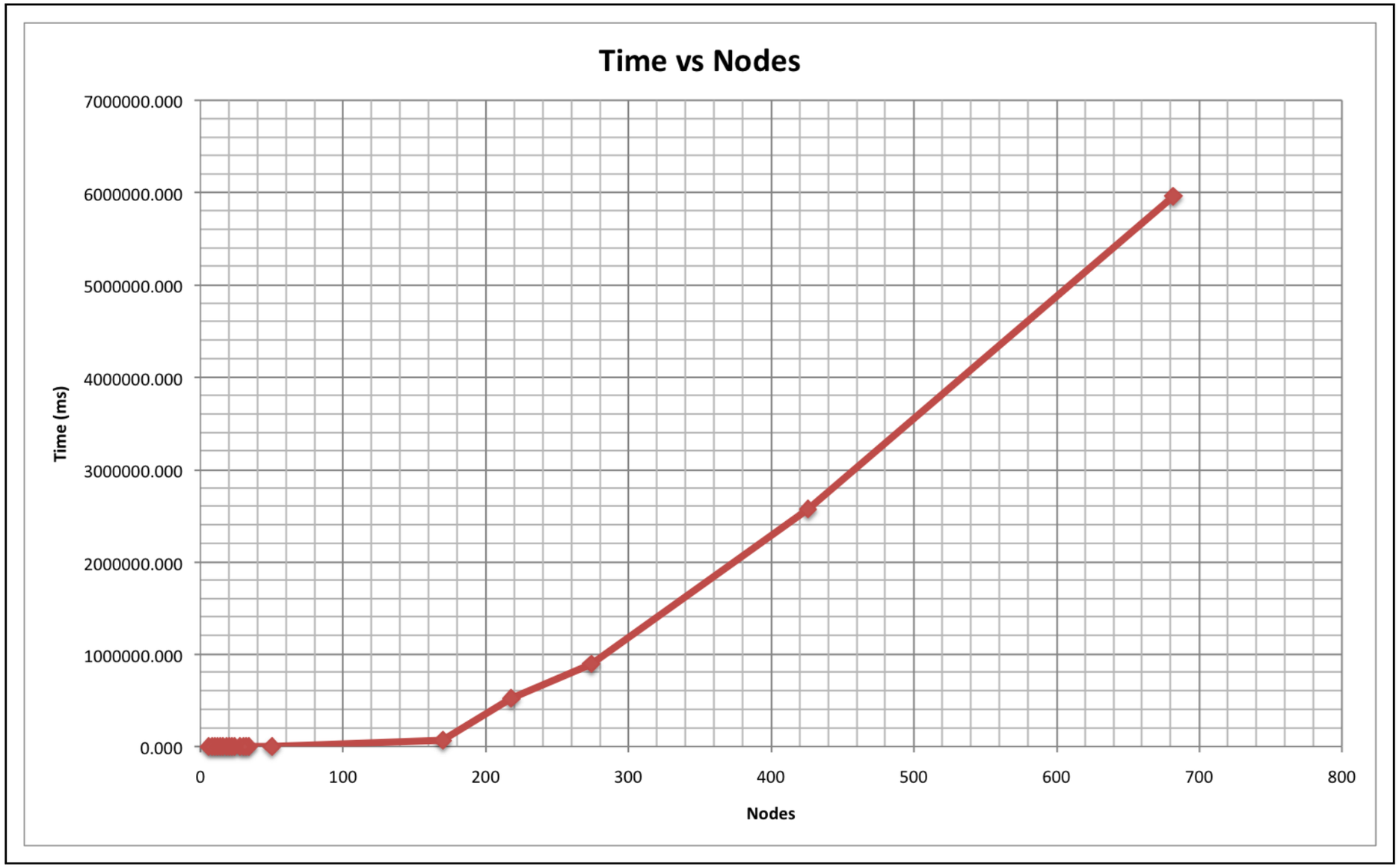


Figure: Time vs. Nodes

Some results: graph 4

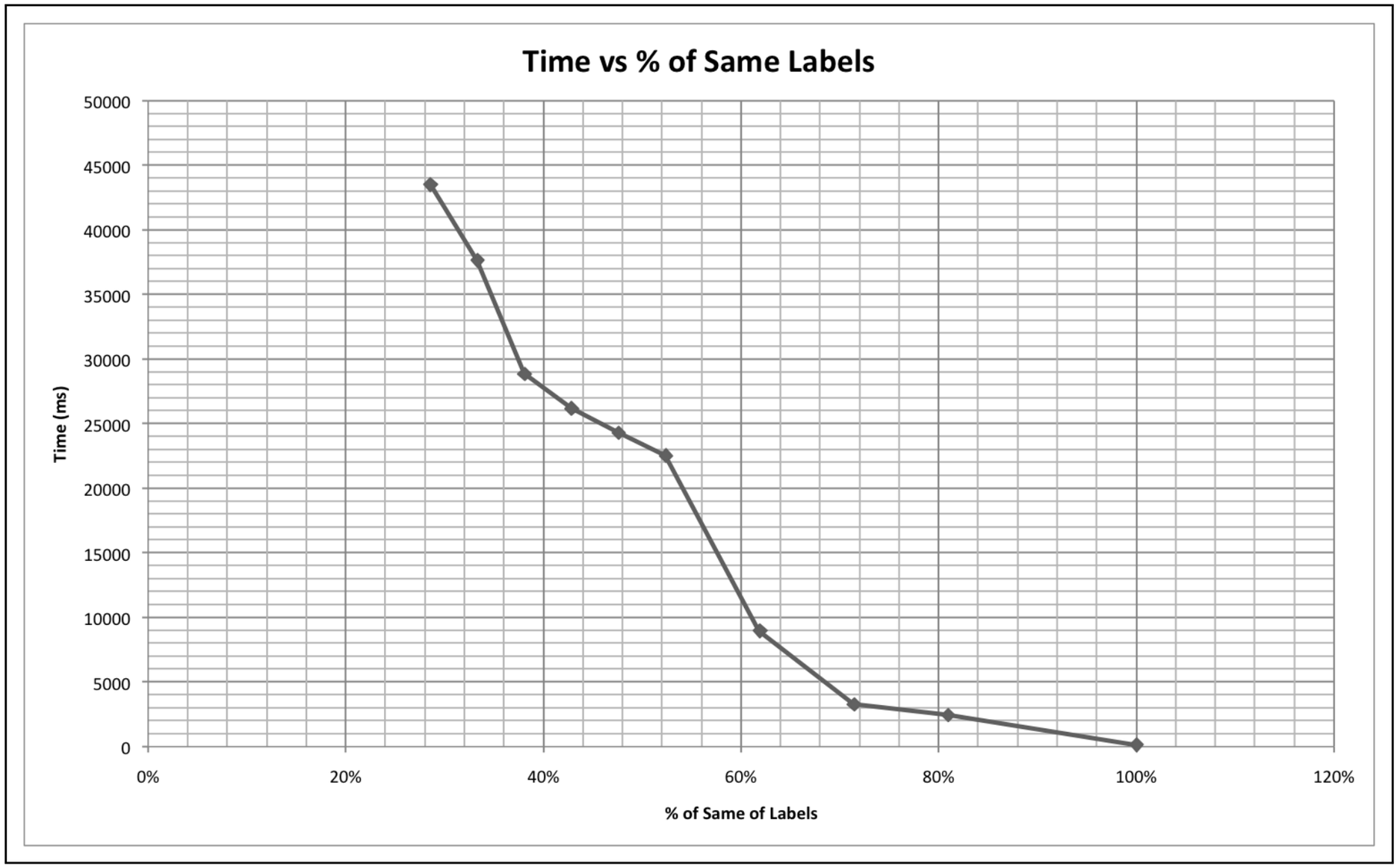


Figure: Time vs Percentage of same labels

Some results: graph 5

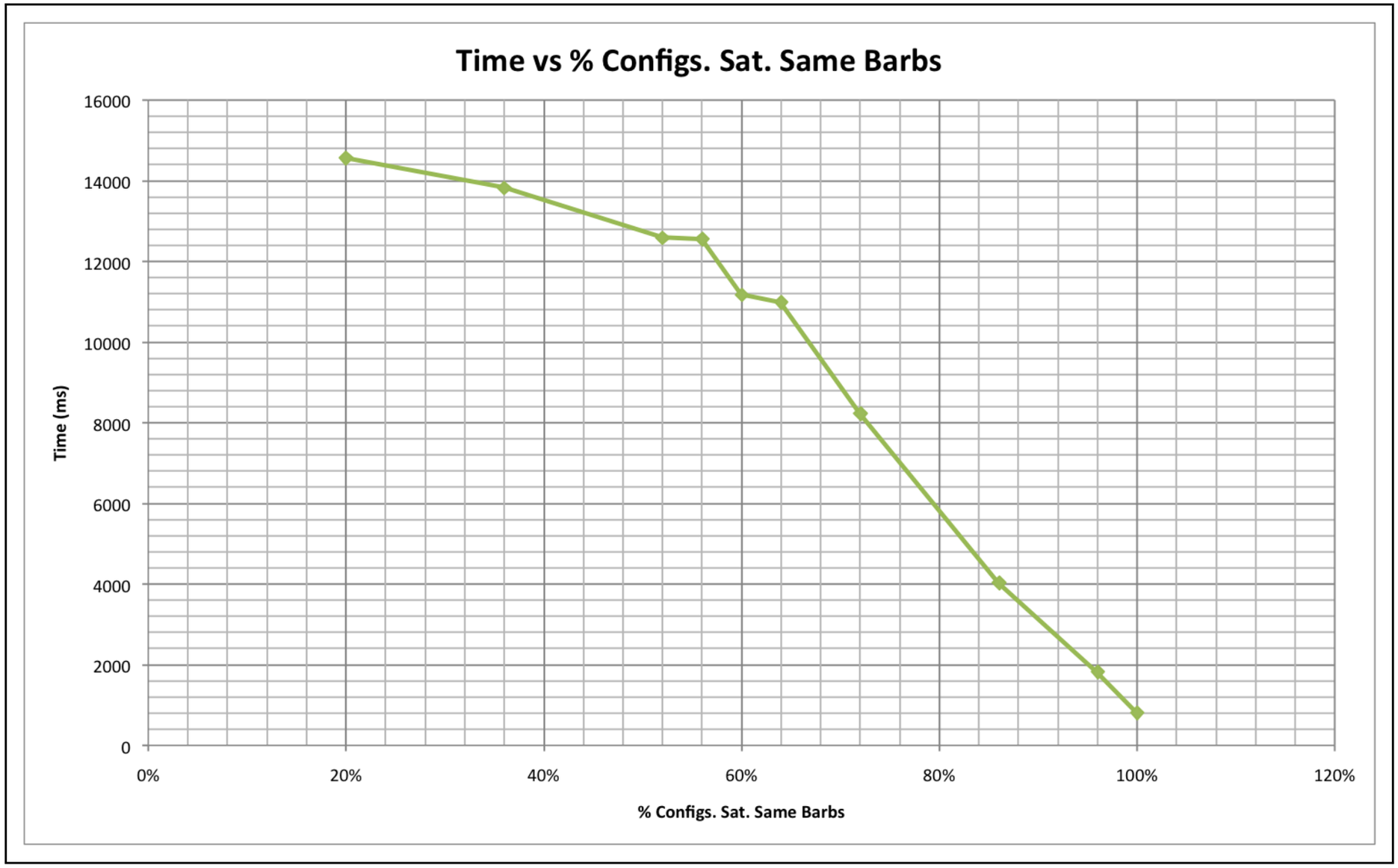


Figure: Time vs Percentage of Configs. Sat. Same Barbs

Some results: graph 6

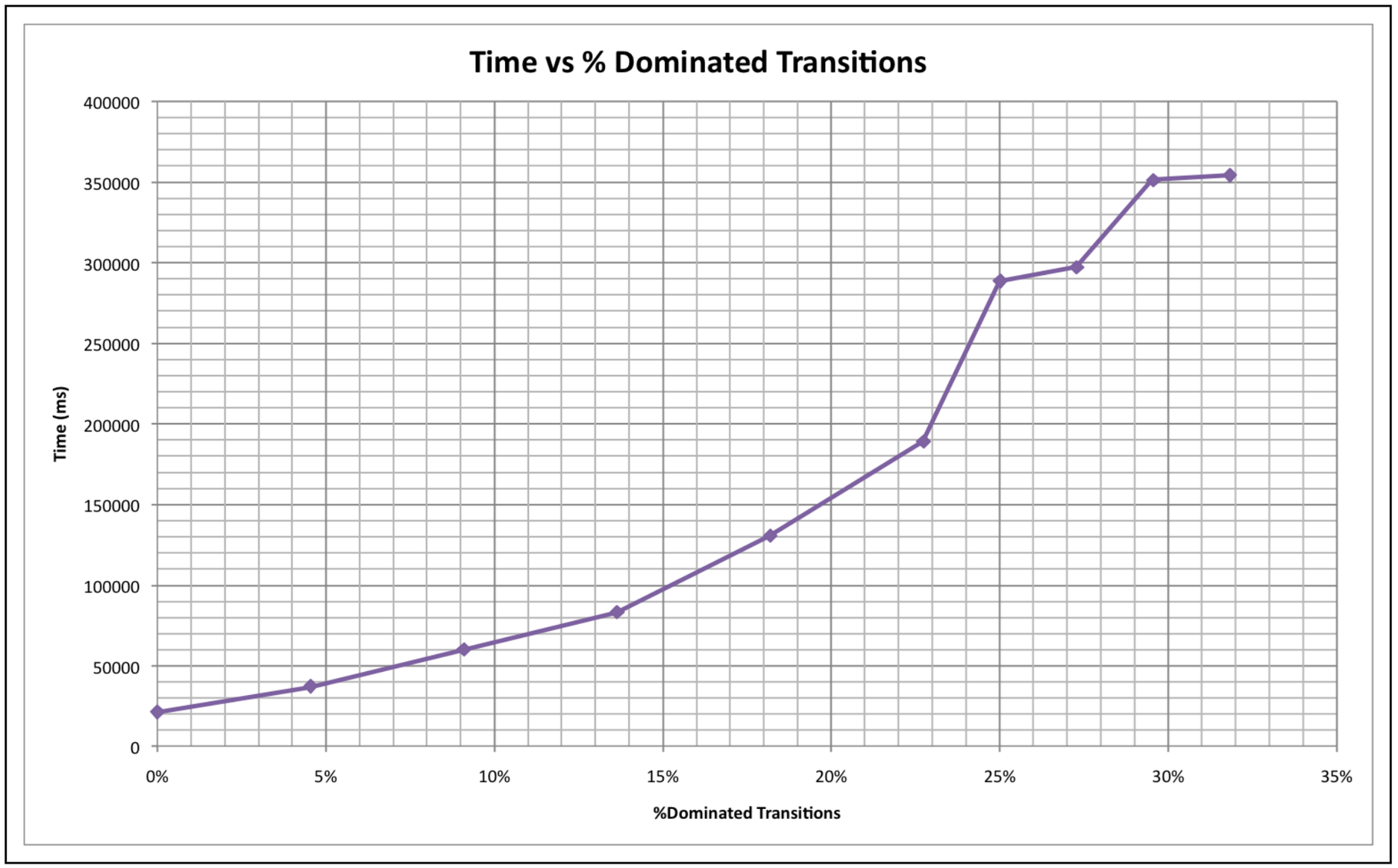


Figure: Time vs Percentage Dominated Transitions

Some results: graph 7

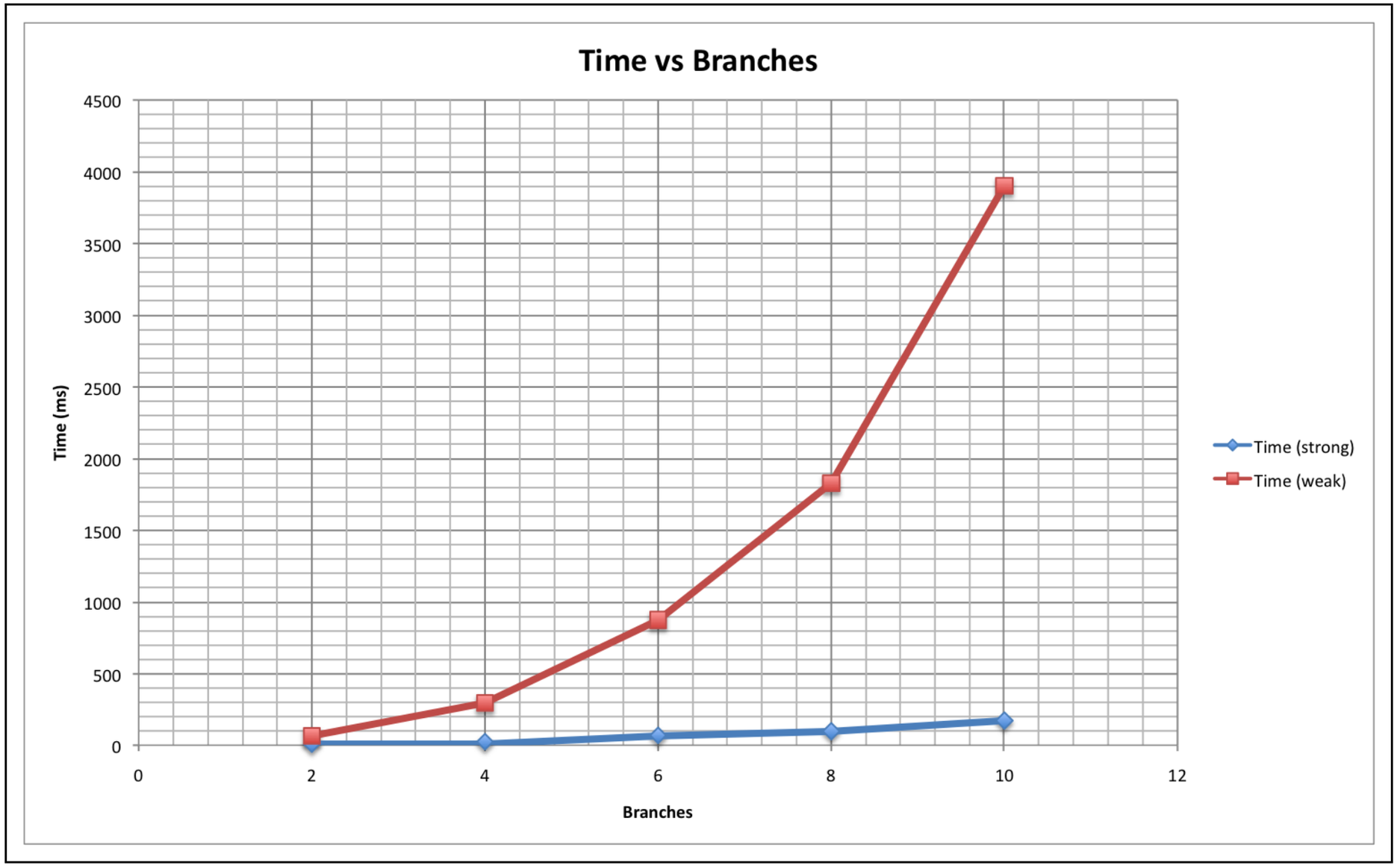


Figure: Time vs Branches

Order theory

- A directed set is a subset A of a partial order set P such that for all pairs $a_1, a_2 \in A$, there exists $a_3 \in A$ with $a_1 \sqsubseteq a_3$ and $a_2 \sqsubseteq a_3$.
- An upper bound of a subset S of some partially ordered set $(P, \sqsubseteq)$ is an element of P which is greater than or equal to every element of S. The term lower bound is defined dually as an element of P which is less than or equal to every element of S. A set with an upper bound is said to be bounded from above by that bound, a set with a lower bound is said to be bounded from below by that bound. The terms bounded above (bounded below) are also used in the mathematical literature for sets that have upper (respectively lower) bounds.
- An element a in a lattice L is compact iff for any directed subset D of L such that $\sqcup D$ exists and $a \sqsubseteq \sqcup D$, then there is an element $d \in D$ such that $a \sqsubseteq d$.

Order theory

- A lattice is a partially ordered set in which any two elements have a supremum (also called a least upper bound or join) and an infimum (also called a greatest lower bound or meet). Lattices can also be characterized as algebraic structures satisfying certain axiomatic identities.
- A poset is called a complete lattice if all its subsets have both a join and a meet. In particular, every complete lattice is a bounded lattice.
- A bounded lattice is an algebraic structure of the form $(L, \sqcup, \sqcap, 1, 0)$ such that $(L, \sqcup, \sqcap)$ is a lattice, 0 (the lattice's bottom) is the identity element for the join operation $\sqcup$, and 1 (the lattice's top) is the identity element for the meet operation $\sqcap$.

Order theory

- There are several notions of "greatest" and "least" element in a poset P, notably:
  - Greatest element and least element: An element $g \in P$ is a greatest element if for every element $a \in P$, $a \sqsubseteq g$. An element $m \in P$ is a least element if for every element $a \in P$, $a \sqsubseteq m$. A poset can only have one greatest or least element.
  - Maximal elements and minimal elements: An element $g \in P$ is a maximal element if there is no element $a \in P$ such that $a \sqsupset g$. Similarly, an element $m \in P$ is a minimal element if there is no element $a \in P$ such that $a \sqsubset m$. If a poset has a greatest element, it must be the unique maximal element, but otherwise there can be more than one maximal element, and similarly for least elements and minimal elements.
- An upper set (also called an upward closed set or just an upset) of a partially ordered set $(X, \sqsubseteq)$ is a subset U with the property that, if x is in U and $x \sqsubseteq y$, then y is in U.

Order theory

- In domain theory, given a partially ordered set, D and a subset, X of D, the upward closure of X in D is the union over all x in X of the sets of all d in D such that $x \sqsubseteq d$. Thus the upward closure of X in D contains the elements of X and any greater element of D. A set is "upward closed" if it is the same as its upward closure, i.e. any d greater than an element is also an element. The downward closure (or "left closure") is similar but with $d \sqsubseteq x$. A downward closed set is one for which any d less than an element is also an element.