

A Solution in Search of a Problem

Formerly: Concise Graphs and Functional
Bisimulations

Ling Cheung and Jesse Hughes

{lcheung,jesseh}@cs.kun.nl.

University of Nijmegen

Overview

- Definitions and Conventions

Overview

- Definitions and Conventions
- Concise Graphs and Least Bisimulation

Overview

- Definitions and Conventions
- Concise Graphs and Least Bisimulation
- Practical Issues

Overview

- Definitions and Conventions
- Concise Graphs and Least Bisimulation
- Practical Issues
- Categorical Constructions

Overview

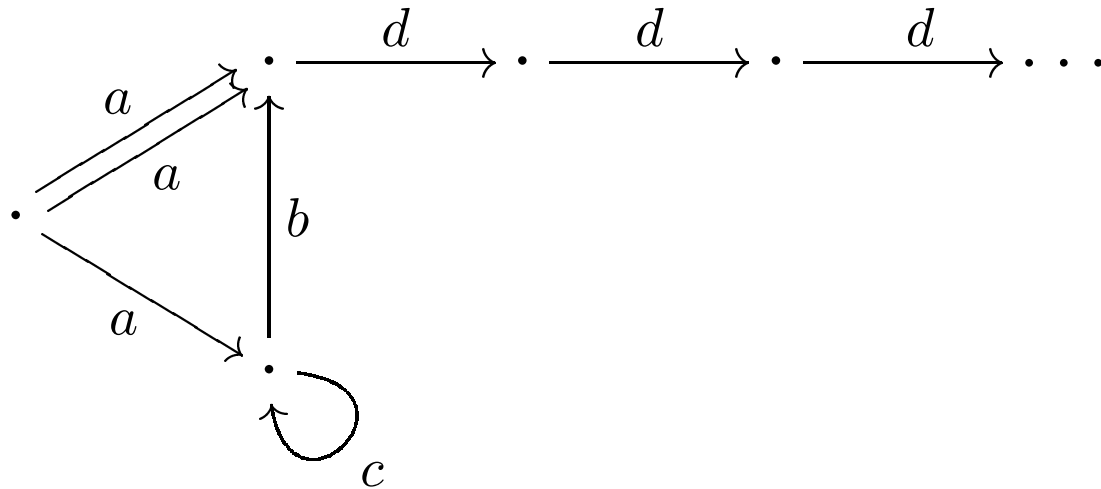
- Definitions and Conventions
- Concise Graphs and Least Bisimulation
- Practical Issues
- Categorical Constructions
- Without Conciseness

Overview

- Definitions and Conventions
- Concise Graphs and Least Bisimulation
- Practical Issues
- Categorical Constructions
- Without Conciseness
- Silent Steps

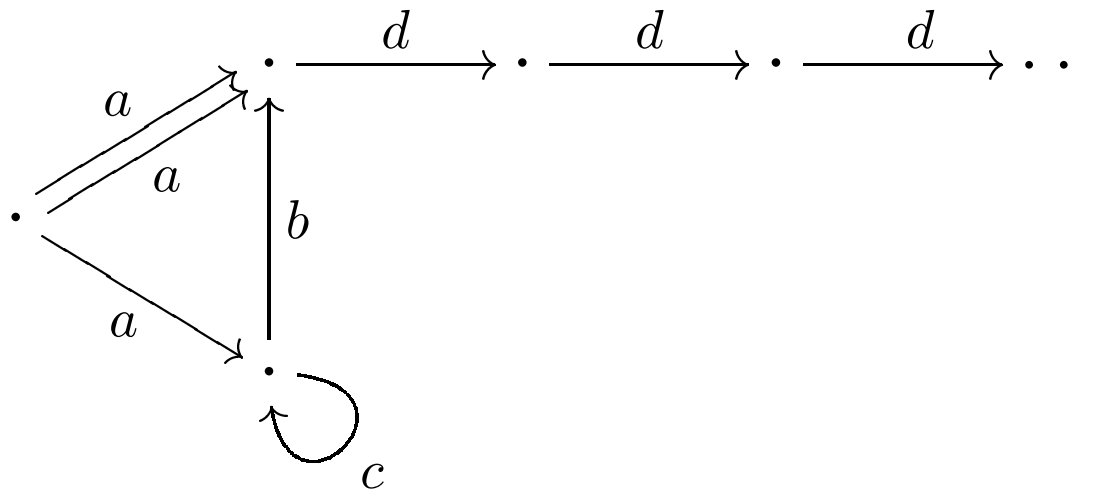
Process Graphs

Graph representation of processes: edge-labeled, directed graphs with multiple roots.



Process Graphs

Graph representation of processes: edge-labeled, directed graphs with multiple roots.



Synonyms: Automata, Labeled Transition Systems (LTS's)

Conventions

A (finite) alphabet \mathcal{A} of atomic actions.

Conventions

A (finite) alphabet \mathcal{A} of atomic actions.

A single step/transition is denoted $s \xrightarrow{a} t$.

Conventions

A (finite) alphabet \mathcal{A} of **atomic actions**.

A single **step/transition** is denoted $s \xrightarrow{a} t$.

A (partial) **run** of the process:

$$r \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n(\dots),$$

where r is a root.

Conventions

A (finite) alphabet \mathcal{A} of **atomic actions**.

A single **step/transition** is denoted $s \xrightarrow{a} t$.

A (partial) **run** of the process:

$$r \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n(\dots),$$

where r is a root.

The **trace** of this run: $a_1 a_2 \dots a_n$.

Process Semantics

A binary relation $R \subseteq G \times H$ is a **bisimulation** if:

- Every root of G is related to some root of H ;

Process Semantics

A binary relation $R \subseteq G \times H$ is a **bisimulation** if:

- Every root of G is related to some root of H ;
- Given $\langle s, t \rangle \in R$ and $s \xrightarrow{a} s'$, there exists t' in H such that $t \xrightarrow{a} t'$ and $\langle s', t' \rangle \in R$;

Process Semantics

A binary relation $R \subseteq G \times H$ is a **bisimulation** if:

- Every root of G is related to some root of H ;
- Given $\langle s, t \rangle \in R$ and $s \xrightarrow{a} s'$, there exists t' in H such that $t \xrightarrow{a} t'$ and $\langle s', t' \rangle \in R$;
- Similarly from H to G .

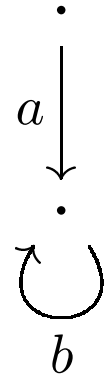
Process Semantics

A binary relation $R \subseteq G \times H$ is a **bisimulation** if:

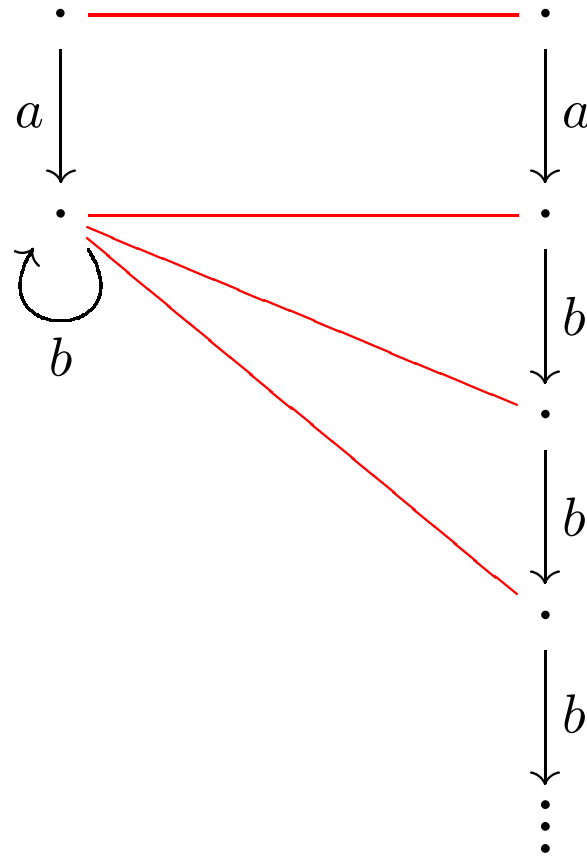
- Every root of G is related to some root of H ;
- Given $\langle s, t \rangle \in R$ and $s \xrightarrow{a} s'$, there exists t' in H such that $t \xrightarrow{a} t'$ and $\langle s', t' \rangle \in R$;
- Similarly from H to G .

We say s is **bisimilar** to t ($s \iff t$) if there is a bisimulation R such that $\langle s, t \rangle \in R$.

Bisimulation example



Bisimulation example



More definitions

A bisimulation R is functional if

$$R = \Phi(f) \text{ for some } f:G \longrightarrow H.$$

More definitions

A bisimulation R is **functional** if

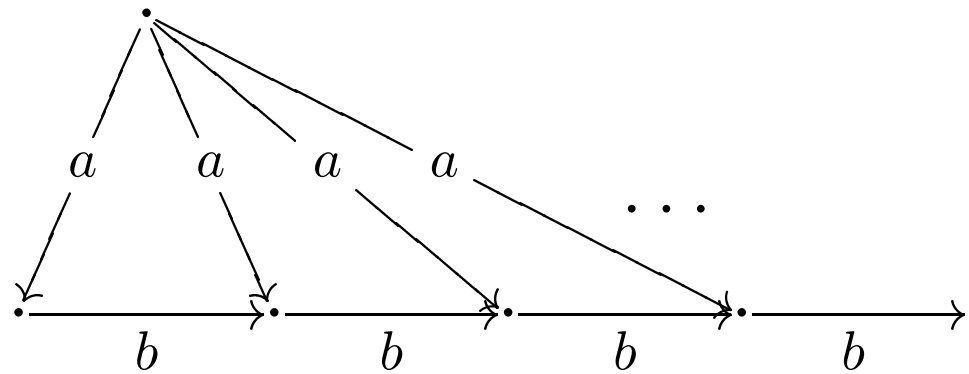
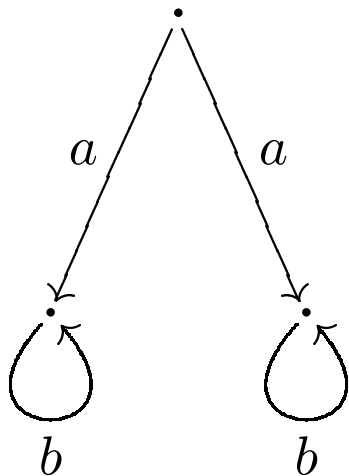
$$R = \Phi(f) \text{ for some } f:G \longrightarrow H.$$

A bisimulation R is **minimal** if

$$R' \subseteq R \text{ and } R' \text{ bisimulation} \Rightarrow R' = R.$$

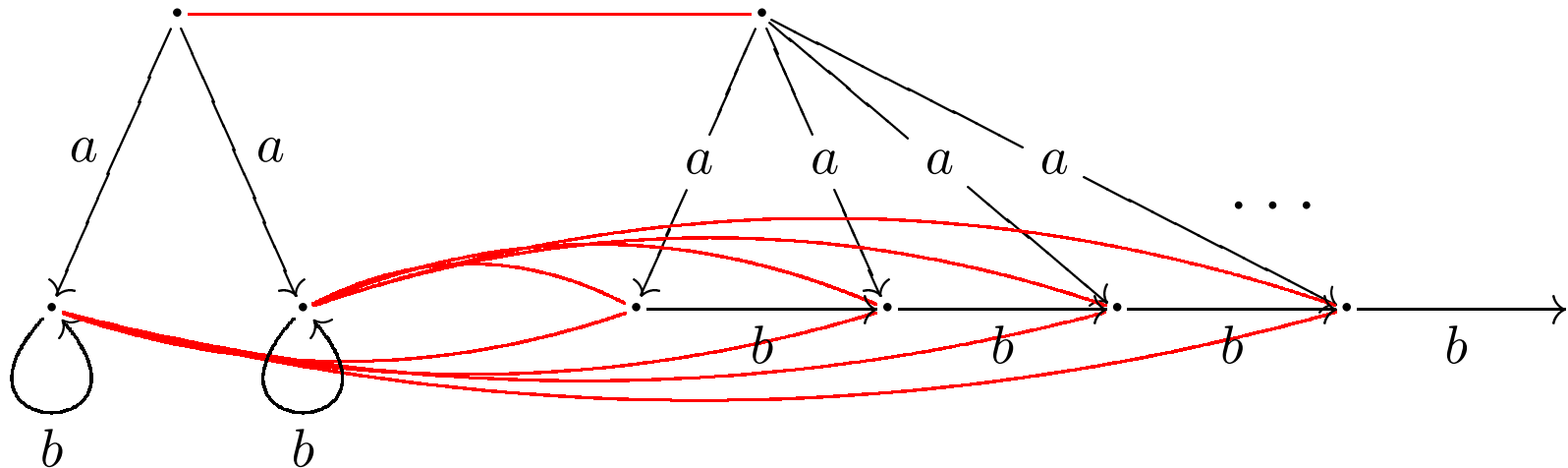
No minimal bisimulation

Consider the following processes.



No minimal bisimulation

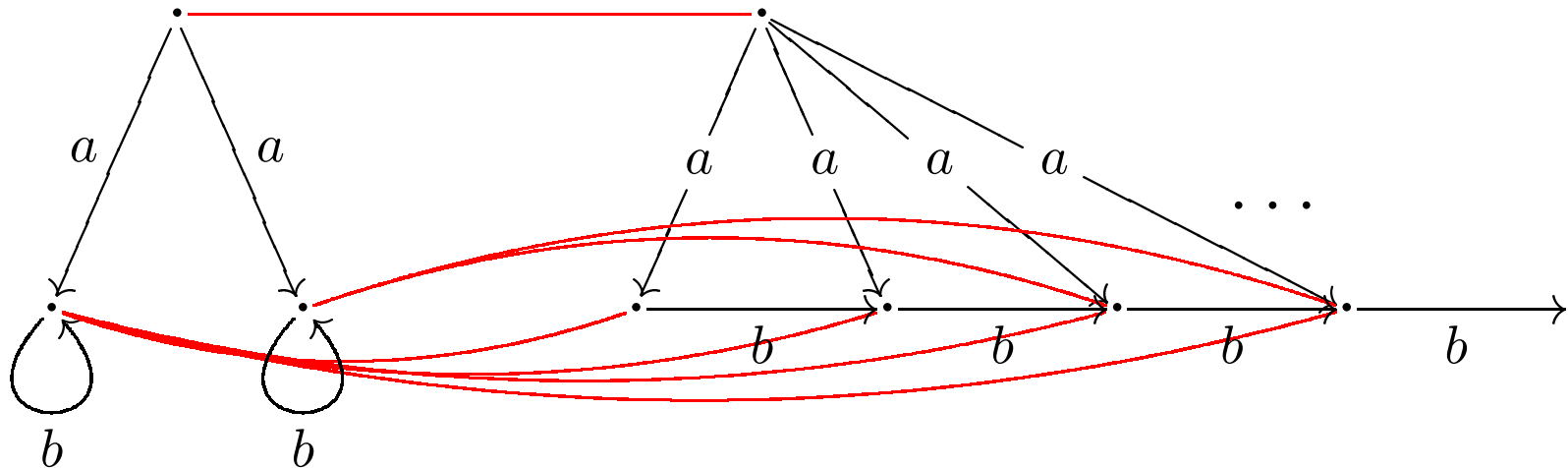
Consider the following processes.



Here is a bisimulation.

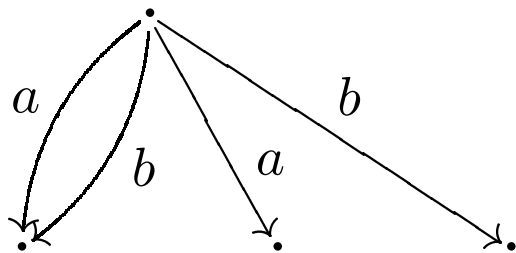
No minimal bisimulation

Consider the following processes.



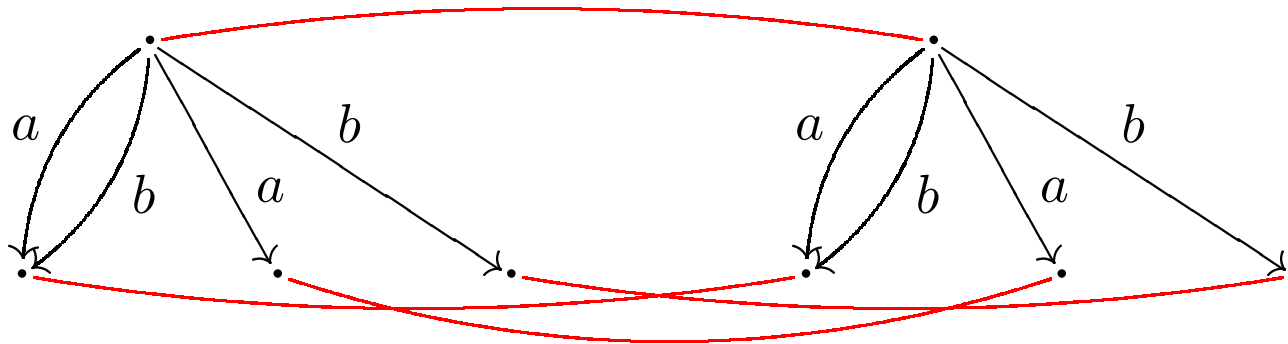
Minimal, not least, bisimulations

Consider the following process.



Minimal, not least, bisimulations

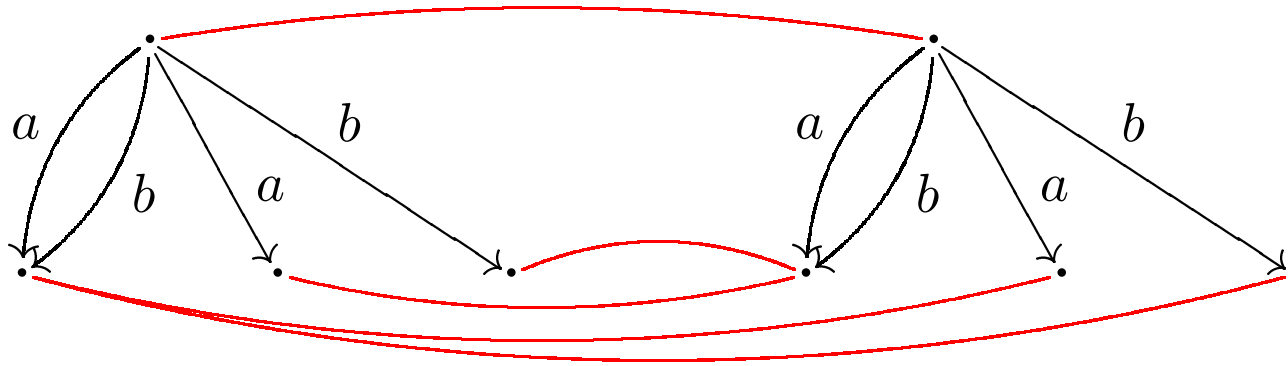
Consider the following process.



Clearly, the identity is a minimal bisimulation.

Minimal, not least, bisimulations

Consider the following process.



This is another minimal bisimulation.

Note: It is not comparable to the identity!

Category of Process Graphs

P: the category of process graphs and functional bisimulations

Category of Process Graphs

P: the category of process graphs and functional bisimulations

Why functional bisimulations?

Category of Process Graphs

P: the category of process graphs and functional bisimulations

Why functional bisimulations?

- We follow Ariola and Klop (1996) on term graphs;

Category of Process Graphs

P: the category of process graphs and functional bisimulations

Why functional bisimulations?

- We follow Ariola and Klop (1996) on term graphs;
- $G \underline{\leftrightarrow} H$ iff there is R with functional bisimulations $g: R \rightarrow G$ and $h: R \rightarrow H$;

Category of Process Graphs

P: the category of process graphs and functional bisimulations

Why functional bisimulations?

- We follow Ariola and Klop (1996) on term graphs;
- $G \underline{\Leftrightarrow} H$ iff there is R with functional bisimulations $g: R \rightarrow G$ and $h: R \rightarrow H$;
- Functional bisimulations \Leftrightarrow (strong) history relations.

Bisimulation as Process Graph

Transition structure on $R \subseteq G \times H$:

- $\langle r_1, r_2 \rangle \in \text{roots}(R)$ iff:

$$r_1 \in \text{roots}(G) \text{ and } r_2 \in \text{roots}(H);$$

Bisimulation as Process Graph

Transition structure on $R \subseteq G \times H$:

- $\langle r_1, r_2 \rangle \in \text{roots}(R)$ iff:

$$r_1 \in \text{roots}(G) \text{ and } r_2 \in \text{roots}(H);$$

- $\langle s, t \rangle \xrightarrow{a} \langle s', t' \rangle$ iff:

$$\begin{array}{c} s \\ \downarrow a \\ s' \end{array}$$

and

$$\begin{array}{c} t \\ \downarrow a \\ t' \end{array}$$

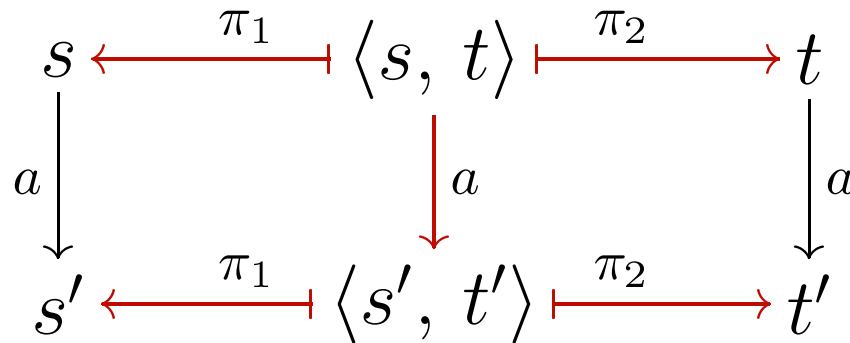
Bisimulation as Process Graph

Transition structure on $R \subseteq G \times H$:

- $\langle r_1, r_2 \rangle \in \text{roots}(R)$ iff:

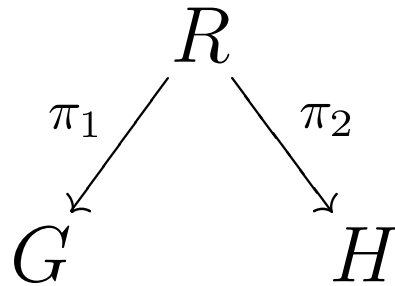
$$r_1 \in \text{roots}(G) \text{ and } r_2 \in \text{roots}(H);$$

- $\langle s, t \rangle \xrightarrow{a} \langle s', t' \rangle$ iff:



Bisimulation as Process Graph

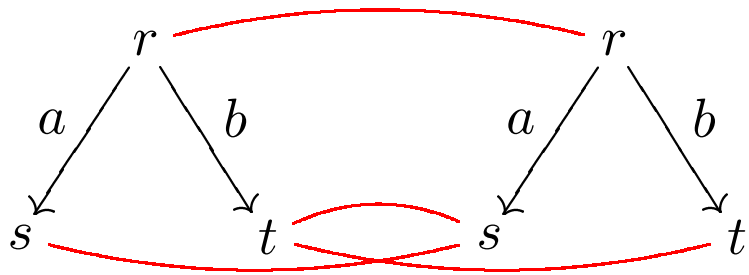
If R is a bisimulation, then



are functional bisimulations.

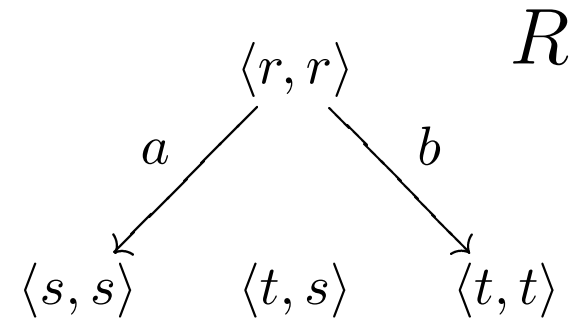
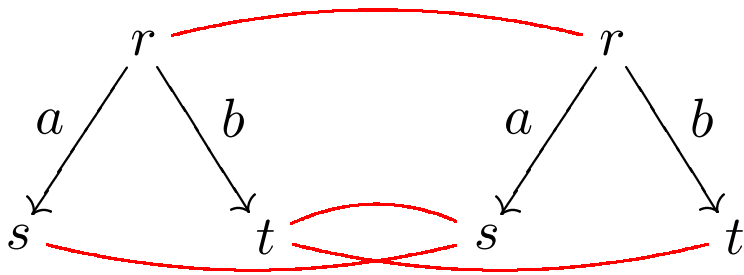
Reachability in R

Example: unreachable node in R .



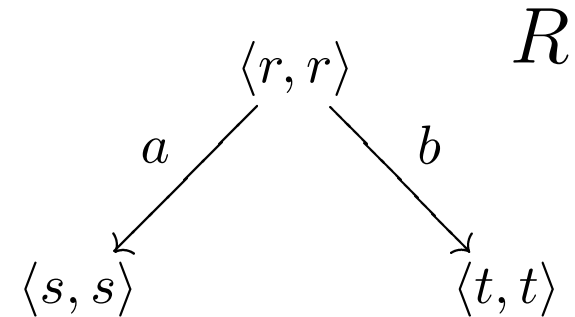
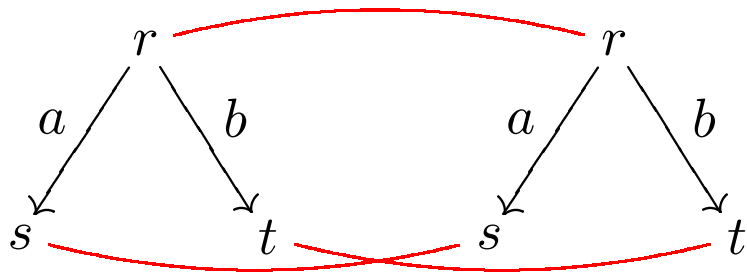
Reachability in R

Example: unreachable node in R .



Reachability in R

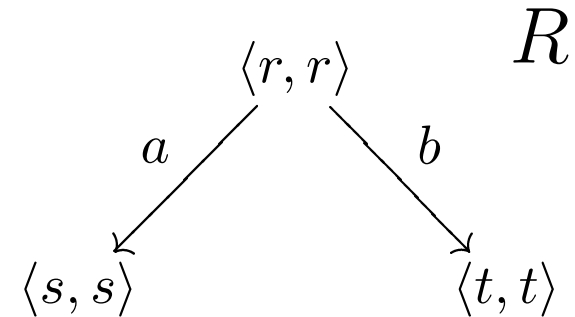
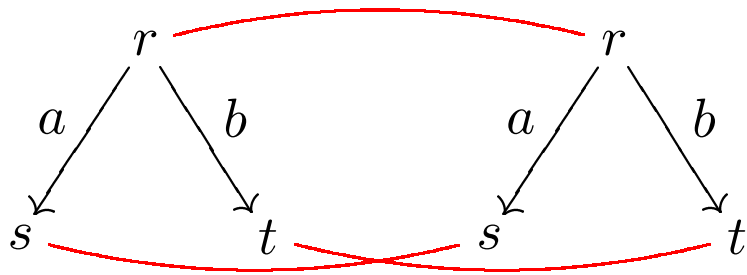
Example: unreachable node in R .



Let $\text{reach}(R)$ denote the reachable part of R .

Reachability in R

Example: unreachable node in R .



Let $\text{reach}(R)$ denote the **reachable part** of R .

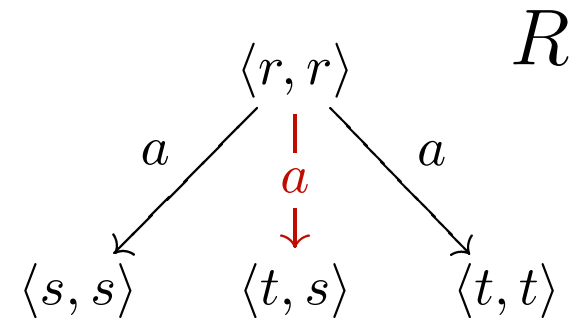
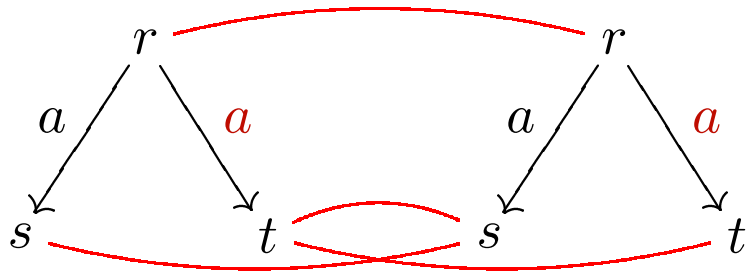
$\text{reach}(R)$ is a bisimulation (so unreachable pairs are “redundant”).

Reachability in R

If R is minimal, then $R = \text{reach}(R)$.

Reachability in R

If R is minimal, then $R = \text{reach}(R)$.
(Converse not true.)



Concise Graphs

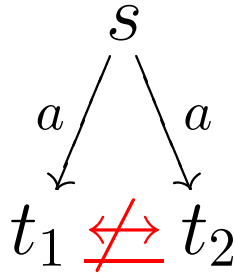
A process graph G is concise if:

- G contains no distinct but bisimilar roots;

Concise Graphs

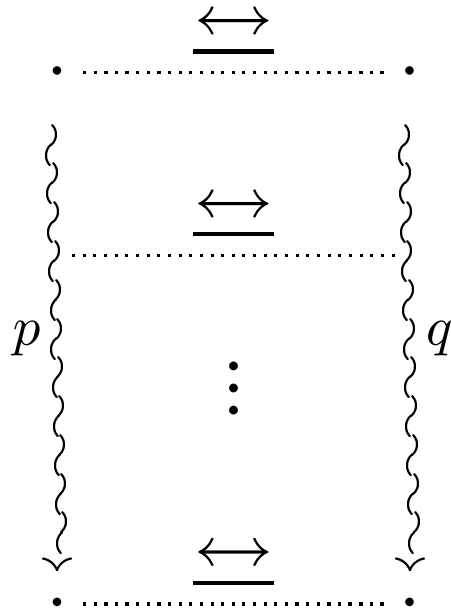
A process graph G is **concise** if:

- G contains no distinct but bisimilar roots;
- **in reach(G):**



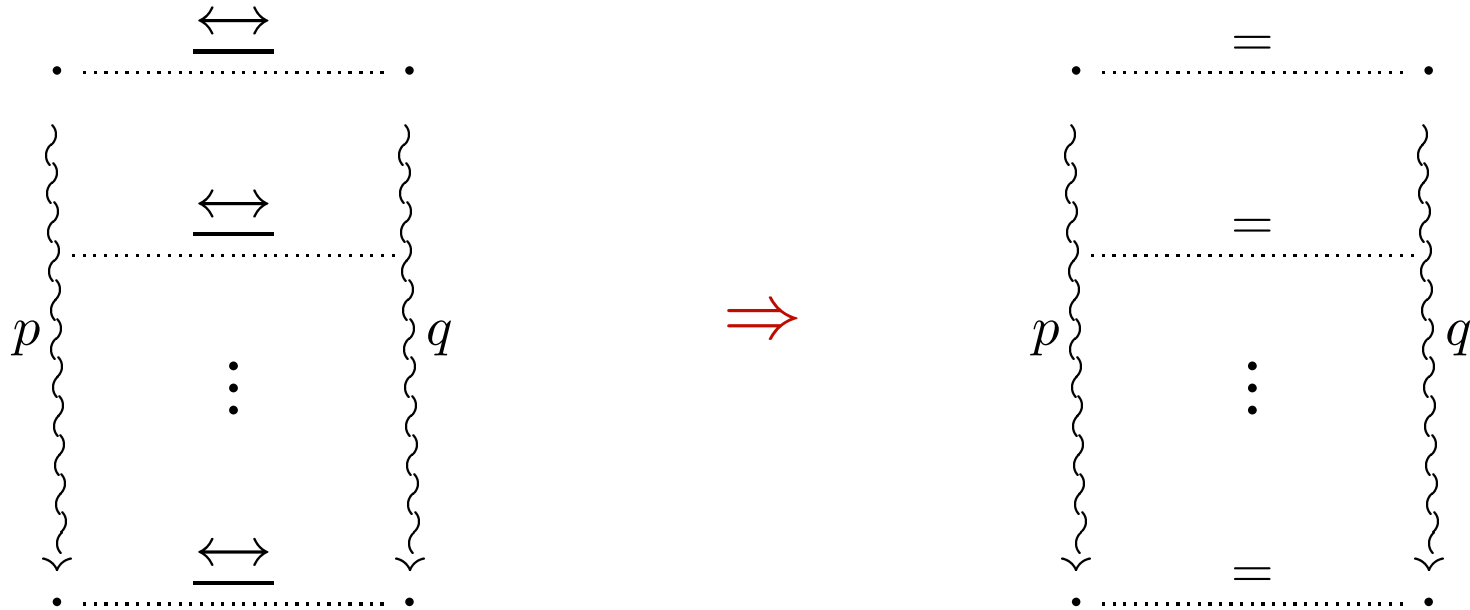
Concise Graphs

Equivalently, given two partial runs p and q ,



Concise Graphs

Equivalently, given two partial runs p and q ,



Existence of Least Bisimulation

Theorem. *G is concise if and only if, for any bisimilar H and any bisimulation R , $\text{reach}(R)$ is the least bisimulation between G and H .*

Existence of Least Bisimulation

Theorem. *G is concise if and only if, for any bisimilar H and any bisimulation R , $\text{reach}(R)$ is the least bisimulation between G and H .*

Remarks:

- Conciseness gives both minimality and uniqueness.

Existence of Least Bisimulation

Theorem. *G is concise if and only if, for any bisimilar H and any bisimulation R , $\text{reach}(R)$ is the least bisimulation between G and H .*

Remarks:

- Conciseness gives both minimality and uniqueness.
- $\text{reach}(R)$ is the intersection of all bisimulations between G and H .

Example: Useless Boolean Test

if *BoolExp* then *A* else *B*

Example: Useless Boolean Test

if BoolExp then A else B

If A and B represent bisimilar states, then this program has a non-concise state graph.

Example: Useless Boolean Test

if BoolExp then A else B

If A and B represent bisimilar states, then this program has a non-concise state graph.

Algorithm to suppress such tests: Fernandez et al. (1995)

Checking Concisenss

Checking conciseness has the same time complexity as checking bisimilarity.

Checking Concisenss

Checking conciseness has the same time complexity as checking bisimilarity.

Modified definition (due to Frits Vaandrager): G is **obviously concise** if

- r_1, r_2 distinct roots $\Rightarrow I(r_1) \neq I(r_2)$;
- in $\text{reach}(G)$,

$$(s \xrightarrow{a} t_1 \text{ and } s \xrightarrow{a} t_2 \text{ and } t_1 \neq t_2) \Rightarrow I(t_1) \neq I(t_2).$$

Note: $I(s)$ denotes initial actions of s .

Checking Concisenss

Checking conciseness has the same time complexity as checking bisimilarity.

Modified definition (due to Frits Vaandrager): G is **obviously concise** if

- r_1, r_2 distinct roots $\Rightarrow I(r_1) \neq I(r_2)$;
- in $\text{reach}(G)$,

$$(s \xrightarrow{a} t_1 \text{ and } s \xrightarrow{a} t_2 \text{ and } t_1 \neq t_2) \Rightarrow I(t_1) \neq I(t_2).$$

Note: $I(s)$ denotes initial actions of s .

Obvious conciseness is a **localized** version of conciseness.

Checking Obvious Concisenss

Linear algorithm to check obvious conciseness?

Checking Obvious Concisenss

Linear algorithm to check obvious conciseness?

- Assume action alphabet \mathcal{A} has a fixed size N .

Checking Obvious Concisenss

Linear algorithm to check obvious conciseness?

- Assume action alphabet \mathcal{A} has a fixed size N .
- Store $I(s)$ as a sorted array.

Checking Obvious Concisenss

Linear algorithm to check obvious conciseness?

- Assume action alphabet \mathcal{A} has a fixed size N .
- Store $I(s)$ as a sorted array.
- Step through state graph and check each node.

Checking Bisimilarity

G deterministic \Rightarrow there is a linear algorithm to check
 $G \underline{\leftrightarrow} H$.

Checking Bisimilarity

G deterministic \Rightarrow there is a linear algorithm to check
 $G \underline{\leftrightarrow} H$.

Similarly for **determinate** process graphs.

Checking Bisimilarity

G deterministic \Rightarrow there is a linear algorithm to check $G \underline{\leftrightarrow} H$.

Similarly for **determinate** process graphs.

Open question: Does (obvious) conciseness provide any improvement to checking bisimilarity?

Checking Bisimilarity

G deterministic \Rightarrow there is a linear algorithm to check $G \underline{\leftrightarrow} H$.

Similarly for **determinate** process graphs.

Open question: Does (obvious) conciseness provide any improvement to checking bisimilarity?

Synchronous product vs. partition refinement.

Coequalizer/Quotient

The category \mathcal{P} has all coequalizers.

Coequalizer/Quotient

The category \mathcal{P} has all coequalizers.

As a consequence, given bisimulation $R \subseteq G \times G$, we can form the quotient process G/R :

- G/R is G/\equiv , where \equiv is the least equivalence relation generated by R ;

Coequalizer/Quotient

The category \mathcal{P} has all coequalizers.

As a consequence, given bisimulation $R \subseteq G \times G$, we can form the quotient process G/R :

- G/R is G/\equiv , where \equiv is the least equivalence relation generated by R ;
- $[r]$ is a root if $r \in \text{roots}(G)$;

Coequalizer/Quotient

The category \mathcal{P} has all coequalizers.

As a consequence, given bisimulation $R \subseteq G \times G$, we can form the quotient process G/R :

- G/R is G/ \equiv , where \equiv is the least equivalence relation generated by R ;
- $[r]$ is a root if $r \in \text{roots}(G)$;
- $[s] \xrightarrow{a} [t]$ if there is $t' \equiv t$ with $s \xrightarrow{a} t'$.

Coequalizer/Quotient

The category \mathcal{P} has all coequalizers.

As a consequence, given bisimulation $R \subseteq G \times G$, we can form the quotient process G/R :

- G/R is G/ \equiv , where \equiv is the least equivalence relation generated by R ;
- $[r]$ is a root if $r \in \text{roots}(G)$;
- $[s] \xrightarrow{a} [t]$ if there is $t' \equiv t$ with $s \xrightarrow{a} t'$.

$$\begin{array}{ccc} R & \rightrightarrows & G \\ & & \downarrow \\ & & G/R \end{array}$$

Coequalizer/Quotient

The category \mathcal{P} has all coequalizers.

As a consequence, given bisimulation $R \subseteq G \times G$, we can form the quotient process G/R :

- G/R is G/\equiv , where \equiv is the least equivalence relation generated by R ;
- $[r]$ is a root if $r \in \text{roots}(G)$;
- $[s] \xrightarrow{a} [t]$ if there is $t' \equiv t$ with $s \xrightarrow{a} t'$.

$$\begin{array}{ccc} R & \rightrightarrows & G \xrightarrow{\text{red}} K \\ & & \downarrow \\ & & G/R \end{array}$$

Coequalizer/Quotient

The category \mathcal{P} has all coequalizers.

As a consequence, given bisimulation $R \subseteq G \times G$, we can form the quotient process G/R :

- G/R is G/\equiv , where \equiv is the least equivalence relation generated by R ;
- $[r]$ is a root if $r \in \text{roots}(G)$;
- $[s] \xrightarrow{a} [t]$ if there is $t' \equiv t$ with $s \xrightarrow{a} t'$.

$$\begin{array}{ccc} R & \rightrightarrows & G & \longrightarrow & K \\ & & \downarrow & & \nearrow \\ & & G/R & & \end{array}$$

Coproducts

If $G \underline{\Leftarrow} H$ and G is concise, then the coproduct of G and H exists in \mathcal{P} .

Coproducts

If $G \underline{\leftrightarrow} H$ and G is concise, then the coproduct of G and H exists in \mathcal{P} .

- disjoint union $G + H$ as a process graph;

Coproducts

If $G \underline{\leftrightarrow} H$ and G is concise, then the coproduct of G and H exists in \mathcal{P} .

- disjoint union $G + H$ as a process graph;
- least bisimulation $R \subseteq G \times H$ gives bisimulation $\overline{R} \subseteq (G + H) \times (G + H)$;

Coproducts

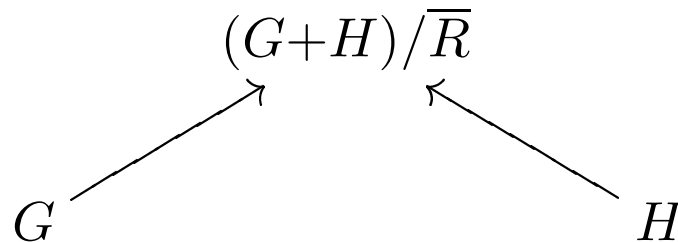
If $G \underline{\leftrightarrow} H$ and G is concise, then the coproduct of G and H exists in \mathcal{P} .

- disjoint union $G + H$ as a process graph;
- least bisimulation $R \subseteq G \times H$ gives bisimulation $\overline{R} \subseteq (G + H) \times (G + H)$;
- coproduct in \mathcal{P} : quotient $(G + H)/\overline{R}$.

Coproduts

If $G \underline{\leftrightarrow} H$ and G is concise, then the coproduct of G and H exists in \mathcal{P} .

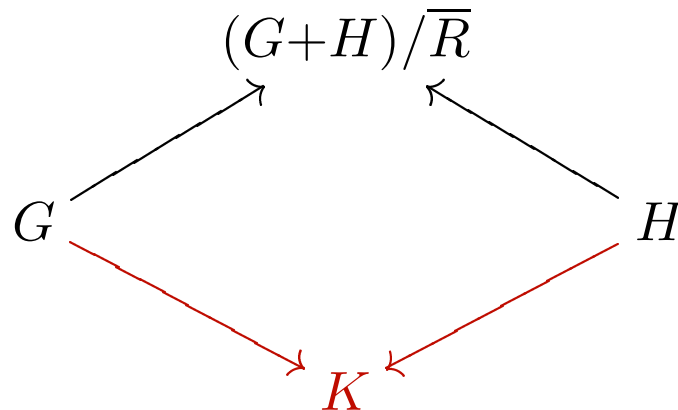
- disjoint union $G + H$ as a process graph;
- least bisimulation $R \subseteq G \times H$ gives bisimulation $\overline{R} \subseteq (G + H) \times (G + H)$;
- coproduct in \mathcal{P} : quotient $(G + H)/\overline{R}$.



Coproducts

If $G \underline{\leftrightarrow} H$ and G is concise, then the coproduct of G and H exists in \mathcal{P} .

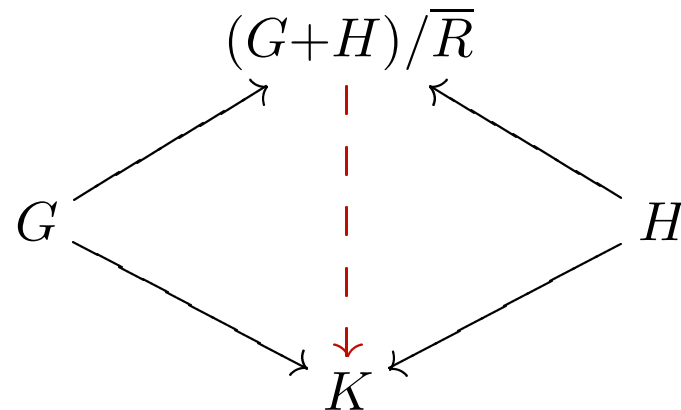
- disjoint union $G + H$ as a process graph;
- least bisimulation $R \subseteq G \times H$ gives bisimulation $\bar{R} \subseteq (G + H) \times (G + H)$;
- coproduct in \mathcal{P} : quotient $(G + H)/\bar{R}$.



Coproducts

If $G \underline{\leftrightarrow} H$ and G is concise, then the coproduct of G and H exists in \mathcal{P} .

- disjoint union $G + H$ as a process graph;
- least bisimulation $R \subseteq G \times H$ gives bisimulation $\bar{R} \subseteq (G + H) \times (G + H)$;
- coproduct in \mathcal{P} : quotient $(G + H)/\bar{R}$.



Products

A graph G is **restricted** if $\text{reach}(G) = G$. Let RP denote the full subcategory of restricted graphs.

Products

A graph G is **restricted** if $\text{reach}(G) = G$. Let RP denote the full subcategory of restricted graphs.

In RP , the product of G and H exists, provided $G \Leftrightarrow H$ and G is concise.

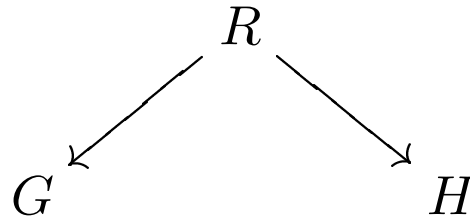
- Take the least bisimulation R .

Products

A graph G is **restricted** if $\text{reach}(G) = G$. Let RP denote the full subcategory of restricted graphs.

In RP , the product of G and H exists, provided $G \underline{\leftrightarrow} H$ and G is concise.

- Take the least bisimulation R .

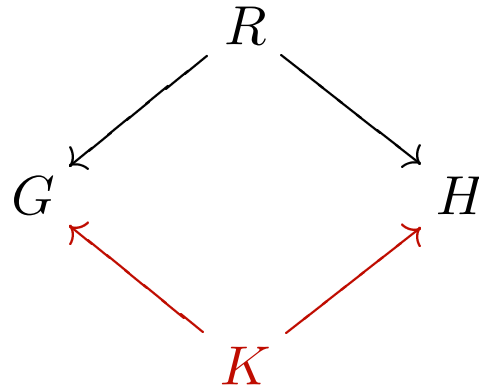


Products

A graph G is **restricted** if $\text{reach}(G) = G$. Let RP denote the full subcategory of restricted graphs.

In RP , the product of G and H exists, provided $G \underline{\leftrightarrow} H$ and G is concise.

- Take the least bisimulation R .

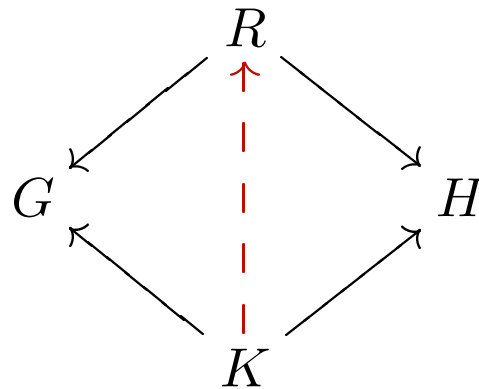


Products

A graph G is **restricted** if $\text{reach}(G) = G$. Let RP denote the full subcategory of restricted graphs.

In RP , the product of G and H exists, provided $G \underline{\leftrightarrow} H$ and G is concise.

- Take the least bisimulation R .



Without Conciseness

G is **image finite** if for all $s \in G$ and for all word σ over \mathcal{A} ,

$\{t \in G \mid s \xrightarrow{\sigma} t\}$ is finite.

Without Conciseness

G is **image finite** if for all $s \in G$ and for all word σ over \mathcal{A} ,

$$\{t \in G \mid s \xrightarrow{\sigma} t\} \text{ is finite.}$$

If G and H are image finite and $G \leftrightarrow H$, then minimal bisimulation exists (but not necessarily unique).

Without Conciseness

Outline of proof:

- Verify that the intersection of a decreasing chain of bisimulations (indexed by the ordinals) is again a bisimulation;

$$R_0 \supseteq \dots \supseteq R_\beta \supseteq R_{\beta+1} \supseteq \dots R_\alpha \dots$$

Without Conciseness

Outline of proof:

- Verify that the intersection of a decreasing chain of bisimulations (indexed by the ordinals) is again a bisimulation;

$$R_0 \supseteq \dots \supseteq R_\beta \supseteq R_{\beta+1} \supseteq \dots R_\alpha \dots$$

This is non-trivial! Requires image finiteness.

Without Conciseness

Outline of proof:

- Verify that the intersection of a decreasing chain of bisimulations (indexed by the ordinals) is again a bisimulation;

$$R_0 \supseteq \dots \supseteq R_\beta \supseteq R_{\beta+1} \supseteq \dots R_\alpha \dots$$

This is non-trivial! Requires image finiteness.

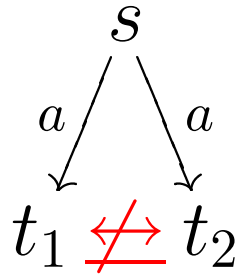
- Apply (well-ordered version of) Zorn's Lemma.

Silent Steps

Modify conciseness to accommodate τ -steps.

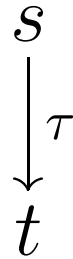
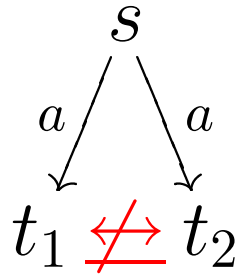
Silent Steps

Modify conciseness to accommodate τ -steps.



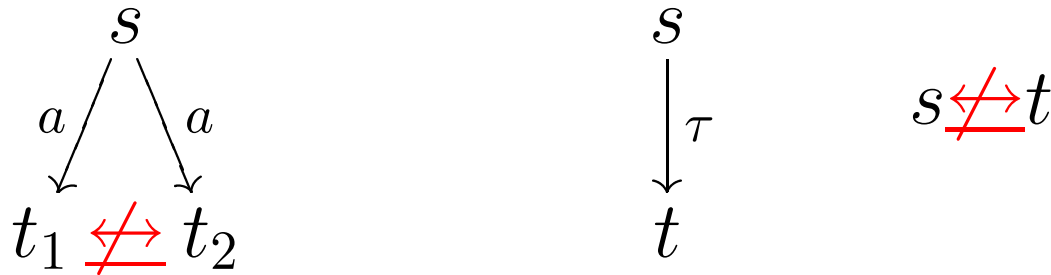
Silent Steps

Modify conciseness to accommodate τ -steps.



Silent Steps

Modify conciseness to accommodate τ -steps.



Consider functional branching bisimulation.