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:

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

where $r$ is a root.
Conventions

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

A single \textbf{step/transition} is denoted \( s \xrightarrow{a} t \).

A (partial) \textbf{run} of the process:

\[
r \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} s_n(\ldots),
\]

where \( r \) is a root.

The \textbf{trace} of this run: \( a_1 a_2 \ldots a_n \).
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 $t$ ($s \leftrightarrow t$) 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

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

A bisimulation $R$ is **functional** if

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

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.
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;
- \( G \leftrightarrow H \) 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;
- \( G \leftrightarrow H \) 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:

  \[ r_1 \in \text{roots}(G) \quad \text{and} \quad r_2 \in \text{roots}(H); \]
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:

\[
\begin{array}{c}
s \\
\downarrow^a \\
s'
\end{array} \quad \text{and} \quad \begin{array}{c}
t \\
\downarrow^a \\
t'
\end{array}
\]
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:

$$
\begin{array}{c}
s \xrightarrow{\pi_1} \langle s, t \rangle \xrightarrow{\pi_2} t \\
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
\langle s', t' \rangle \xrightarrow{\pi_1} \langle s', t' \rangle \xrightarrow{\pi_2} t'
\end{array}
$$
Bisimulation as Process Graph

If $R$ is a bisimulation, then

are functional bisimulations.
Reachability in $R$

Example: unreachable node in $R$. 

$\begin{array}{c}
\text{r} \\
\text{a} \\
\text{s}
\end{array}
\begin{array}{c}
\text{b} \\
\text{t}
\end{array}
\begin{array}{c}
\text{a} \\
\text{s}
\end{array}
\begin{array}{c}
\text{b} \\
\text{t}
\end{array}$
Reachability in $\mathcal{R}$

Example: unreachable node in $\mathcal{R}$.
Reachability in $\mathcal{R}$

Example: unreachable node in $\mathcal{R}$.

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

$$\begin{array}{c}
S \\
\downarrow a \\
t_1 \leftrightarrow t_2 \\
\downarrow a
\end{array}$$
Concise Graphs

Equivalently, given two partial runs $p$ and $q$,
Concise Graphs

Equivalently, given two partial runs $p$ and $q$, 

\[ \begin{array}{c}
\begin{array}{c}
\text{\tikz[baseline=-0.5ex]	ikzstyle{every picture}+=[inner sep=0,outer sep=0]
\begin{scope}
\draw[rounded corners, dotted]
(0,0) rectangle (1,1);
\end{scope}}
\end{array}
\end{array}\]

$\Leftrightarrow$

\[ \begin{array}{c}
\begin{array}{c}
\text{\tikz[baseline=-0.5ex]	ikzstyle{every picture}+=[inner sep=0,outer sep=0]
\begin{scope}
\draw[rounded corners, dotted]
(0,0) rectangle (1,1);
\end{scope}}
\end{array}
\end{array}\]

$\Rightarrow$

\[ \begin{array}{c}
\begin{array}{c}
\text{\tikz[baseline=-0.5ex]	ikzstyle{every picture}+=[inner sep=0,outer sep=0]
\begin{scope}
\draw[rounded corners, dotted]
(0,0) rectangle (1,1);
\end{scope}}
\end{array}
\end{array}\]

$\Leftrightarrow$

\[ \begin{array}{c}
\begin{array}{c}
\text{\tikz[baseline=-0.5ex]	ikzstyle{every picture}+=[inner sep=0,outer sep=0]
\begin{scope}
\draw[rounded corners, dotted]
(0,0) rectangle (1,1);
\end{scope}}
\end{array}
\end{array}\]

$\Rightarrow$

\[ \begin{array}{c}
\begin{array}{c}
\text{\tikz[baseline=-0.5ex]	ikzstyle{every picture}+=[inner sep=0,outer sep=0]
\begin{scope}
\draw[rounded corners, dotted]
(0,0) rectangle (1,1);
\end{scope}}
\end{array}
\end{array}\]
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

\[
\text{if } \text{BoolExp} \text{ then } A \text{ else } B
\]

If \( A \) and \( B \) represent bisimilar states, then this program has a non-concise state graph.
Example: Useless Boolean Test

if $\text{BoolExp}$ 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 Conciseness

Checking conciseness has the same time complexity as checking bisimilarity.
Checking Conciseness

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 Conciseness

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

\[(s \xrightarrow{a} t_1 \text{ and } s \xrightarrow{a} t_2 \text{ and } t_1 \neq t_2) \implies I(t_1) \neq I(t_2).\]

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

Obvious conciseness is a localized version of conciseness.
Checking Obvious Conciseness

Linear algorithm to check obvious conciseness?
Checking Obvious Conciseness

Linear algorithm to check obvious conciseness?

- Assume action alphabet $A$ has a fixed size $N$. 
Checking Obvious Conciseness

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 Conciseness

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 $G \leftrightarrow H$. 
Checking Bisimilarity

\[ G \text{ deterministic} \Rightarrow \text{there is a linear algorithm to check} \]
\[ G \leftrightarrow H. \]

Similarly for \text{determinate} process graphs.
Checking Bisimilarity

$G$ deterministic $\implies$ there is a linear algorithm to check $G \leftrightarrow H$.

Similarly for determinate process graphs.

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

\[ G \text{ deterministic} \Rightarrow \text{there is a linear algorithm to check} \]

\[ G \leftrightarrow H. \]

Similarly for \textit{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 $P$ 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 $s \xrightarrow{a} t'$.
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 $s \xrightarrow{a} t'$.

\[
\begin{array}{c}
\xrightarrow{R} \quad \xrightarrow{a} \quad \xrightarrow{G} \\
\downarrow \\
G/R
\end{array}
\]
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 $s \xrightarrow{a} t'$.

$$
\begin{array}{ccc}
R & \xrightarrow{\sim} & G \\
\downarrow & & \downarrow \\
G/R & \xrightarrow{\sim} & K
\end{array}
$$
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 $r \in \text{roots}(G)$;
- $[s] \xrightarrow{\alpha} [t]$ if there is $t' \equiv t$ with $s \xrightarrow{\alpha} t'$.
Coprodusts

If $G \leftrightarrow H$ and $G$ is concise, then the coproduct of $G$ and $H$ exists in $P$. 
Coproducts

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

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

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

- 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 $P$: quotient $(G + H)/\overline{R}$. 


Coproducts

If $G \leftrightarrow H$ 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 $(G + H)/\overline{R}$.
If $G \equiv H$ and $G$ is concise, then the coproduct of $G$ and $H$ exists in $P$.

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

If $G \leftrightarrow H$ 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 $(G + H)/\overline{R}$.
Products

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

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

In $\text{RP}$, 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 $\text{RP}$ denote the full subcategory of restricted graphs.

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

- Take the least bisimulation $R$. 

```
R

G

H
```
Products

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

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

- Take the least bisimulation $R$. 

\[
\begin{array}{c}
\text{G} \\
\text{R}
\end{array}
\quad
\begin{array}{c}
\text{H} \\
\text{R}
\end{array}
\]

\[
\begin{array}{c}
\text{K} \\
\text{G}
\end{array}
\quad
\begin{array}{c}
\text{K} \\
\text{H}
\end{array}
\]
Products

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

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

- Take the least bisimulation $R$. 

\[
\begin{array}{c}
G \\
\downarrow \quad R \\
K \\
\uparrow \\
H \\
\end{array}
\]
Without Conciseness

$G$ is **image finite** if for all $s \in G$ and for all word $\sigma$ over $A$,

$$\{ t \in G \mid s \xrightarrow{\sigma} t \}$$

is finite.
Without Conciseness

$G$ is image finite if for all $s \in G$ and for all word $\sigma$ 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;

\[ R_0 \supseteq \ldots \supseteq R_\beta \supseteq R_{\beta+1} \supseteq \ldots \supseteq R_\alpha \ldots \]
Without Conciseness

Outline of proof:

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

\[ R_0 \supseteq \ldots \supseteq R_\beta \supseteq R_{\beta+1} \supseteq \ldots R_\alpha \ldots \]

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;

\[ R_0 \supseteq \ldots \supseteq R_\beta \supseteq R_{\beta + 1} \supseteq \ldots R_\alpha \ldots \]

This is non-trivial! Requires image finiteness.

- Apply (well-ordered version of) Zorn’s Lemma.
Silent Steps

Modify conciseness to accommodate $\tau$-steps.
Silent Steps

Modify conciseness to accommodate $\tau$-steps.

$S \xleftarrow{a} t_1 \not\equiv t_2 \xrightarrow{a} S$
Silent Steps

Modify conciseness to accommodate $\tau$-steps.

\[
\begin{array}{c}
S \\
\downarrow a \\
t_1 \leftrightarrow t_2 \\
\downarrow a \\
S \\
\downarrow \tau \\
t \\
\end{array}
\]

$s \not\leftrightarrow t$
Silent Steps

Modify conciseness to accommodate $\tau$-steps.

Consider functional branching bisimulation.