Bisimulation Slices and Transfer Functions

Clemens Grabmayer

Department of Computer Science

GRAN SASSO SCIENCE INSTITUTE
SCHOOL OF ADVANCED STUDIES
Scuola Universitaria Superiore

L'Aquila, Italy

NWPT 2021
Reykjavik, Iceland, and online
4–6 November, 2021

Overview

- fragments of bisimulations
  - motivation 1: nearly collapsed LTSs (labeled transition systems) for the process semantics of regular expressions
  - bisimulating slices and grounded bisimulation slices
- functional bisimulation fragments
  - motivation 2: transfer of LTS spec's via functional bisimulations
  - transfer functions and local transfer functions
  - transfer functions via elevation from local-transfer functions
- application aimed at
  - using transfer to prove specifications equal on nearly collapsed 1-LTSs

Overview

- fragments of bisimulations
  - motivation 1: nearly collapsed LTSs (labeled transition systems) for the process semantics of regular expressions
  - bisimulating slices and grounded bisimulation slices
- functional bisimulation fragments
  - motivation 2: transfer of LTS spec's via functional bisimulations
  - transfer functions and local transfer functions
  - transfer functions via elevation from local-transfer functions
- application aimed at
  - using transfer to prove specifications equal on nearly collapsed 1-LTSs

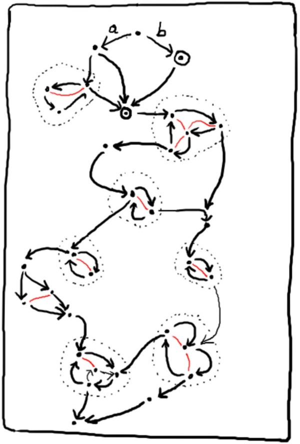
Overview

- fragments of bisimulations
  - motivation 1: nearly collapsed LTSs (labeled transition systems) for the process semantics of regular expressions
  - bisimulating slices and grounded bisimulation slices
- functional bisimulation fragments
  - motivation 2: transfer of LTS spec's via functional bisimulations
  - transfer functions and local transfer functions
  - transfer functions via elevation from local-transfer functions
- application aimed at
  - using transfer to prove specifications equal on nearly collapsed 1-LTSs

Motivation 1: nearly collapsed LTSs

strongly connected components (scc's)

termination permitted

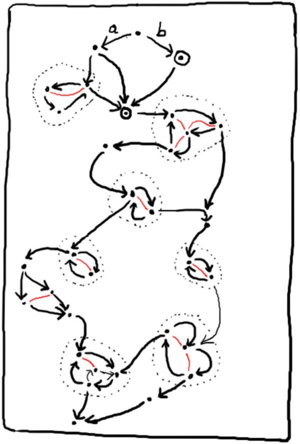


— bisimilarity is contained within scc's

Motivation 1: nearly collapsed LTSs

strongly connected components (scc's)

termination permitted



— bisimilarity

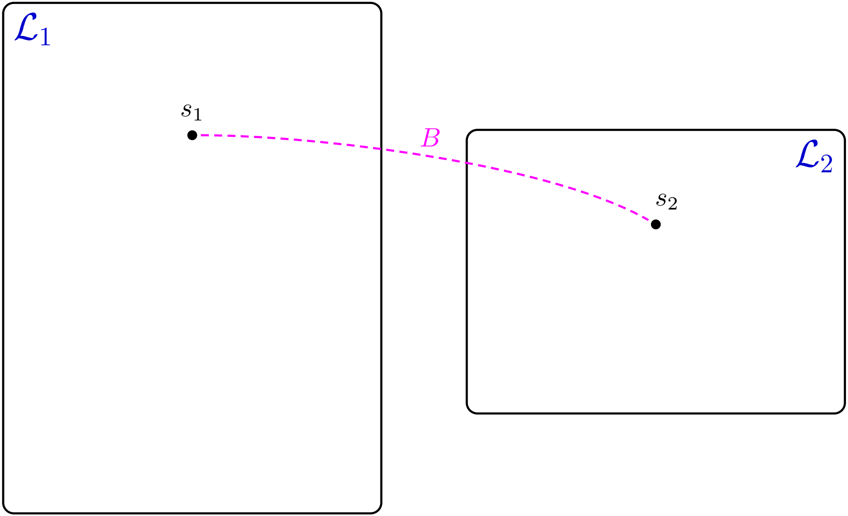
is contained within scc's

Definition

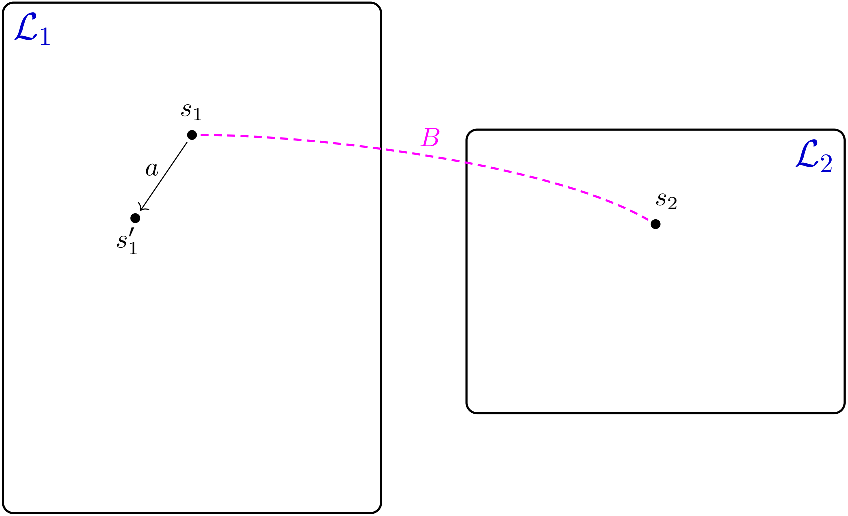
A *labeled transition system* is a tuple $\mathcal{L} = \langle S, A, \to, \downarrow \rangle$ where:

- S is a set of states,
- A is a set of actions,
- $\to \subseteq S \times A \times S$ is a set of labeled transitions,
- $\downarrow \subseteq V$ is a set of terminating states.

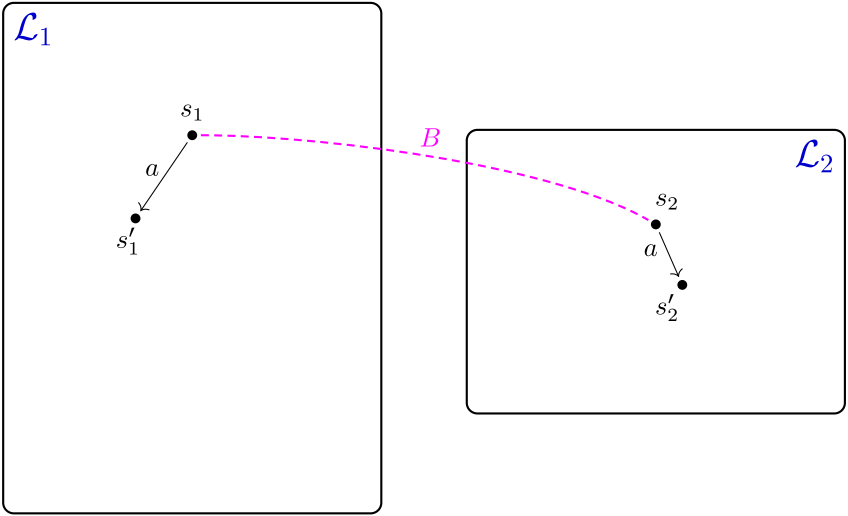
Bisimulation



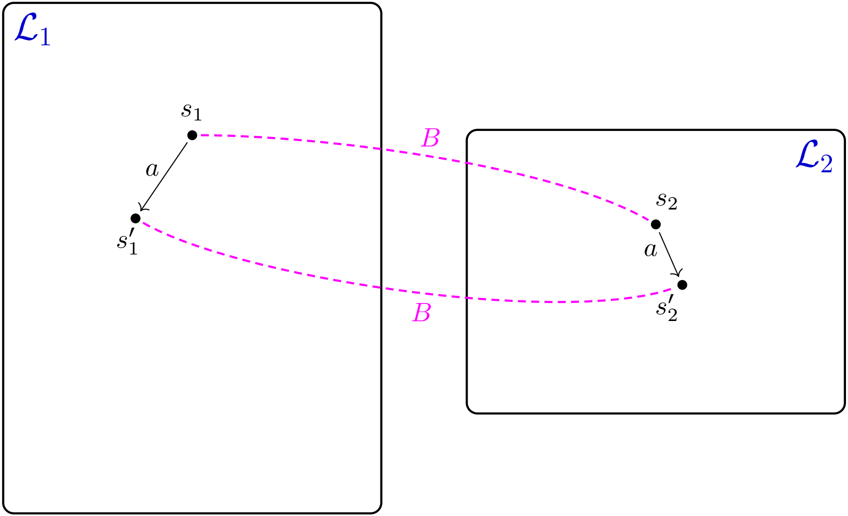
Bisimulation (forth condition)



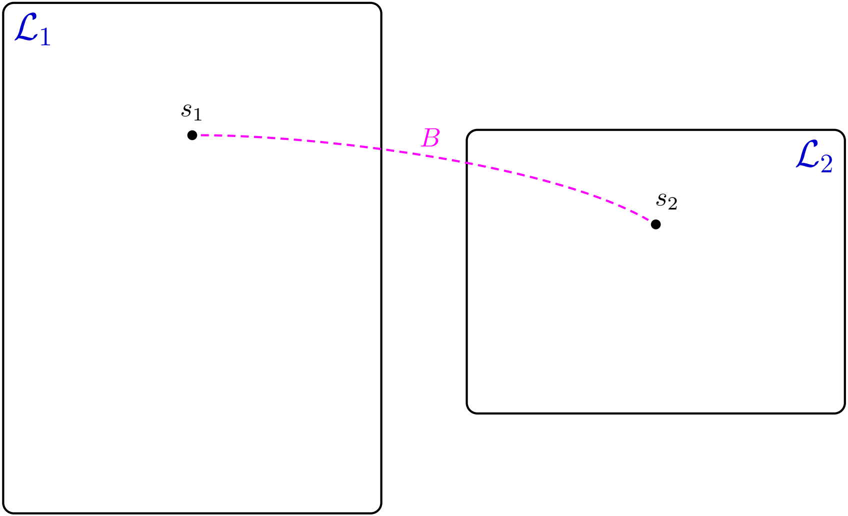
Bisimulation (forth condition)



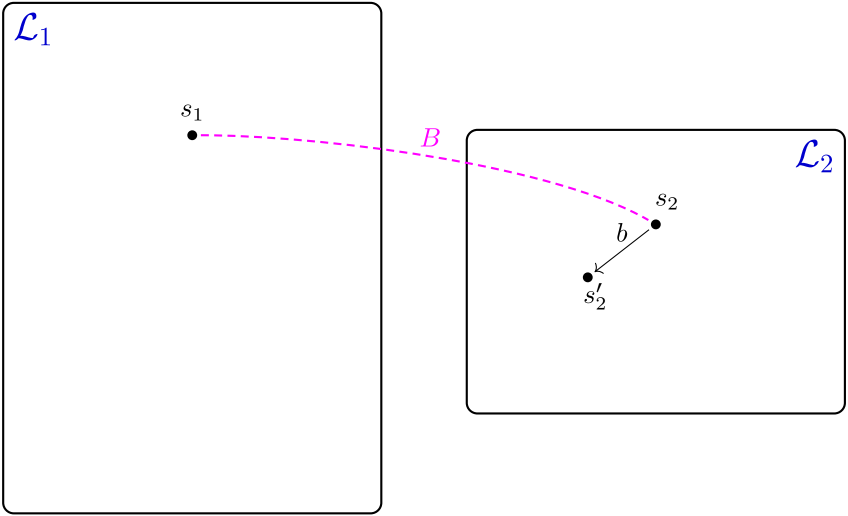
Bisimulation (forth condition)



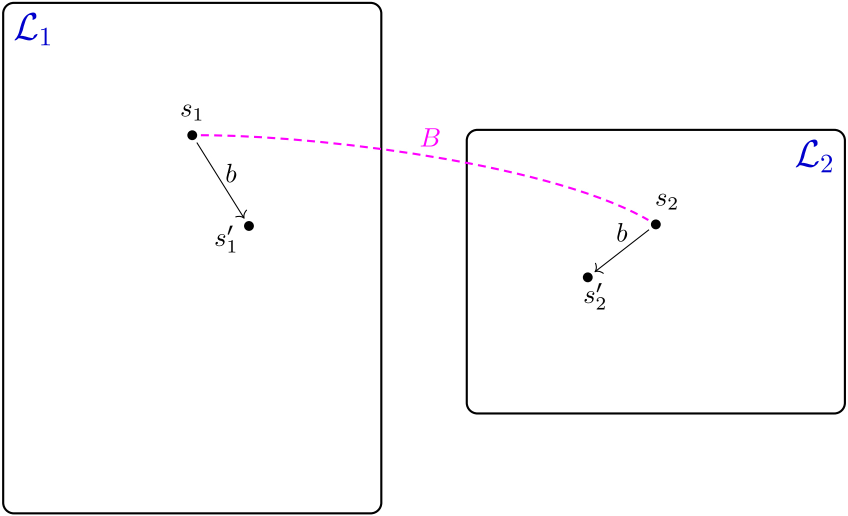
Bisimulation (forth condition)



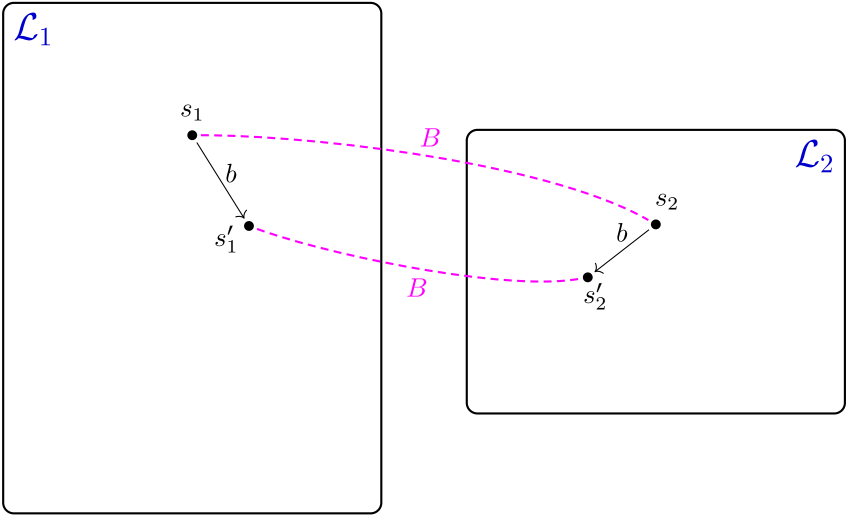
Bisimulation (back condition)



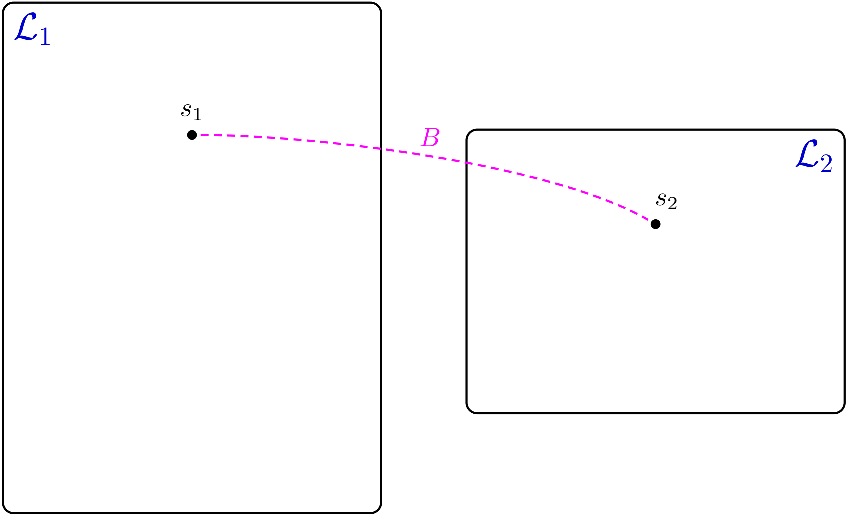
Bisimulation (back condition)



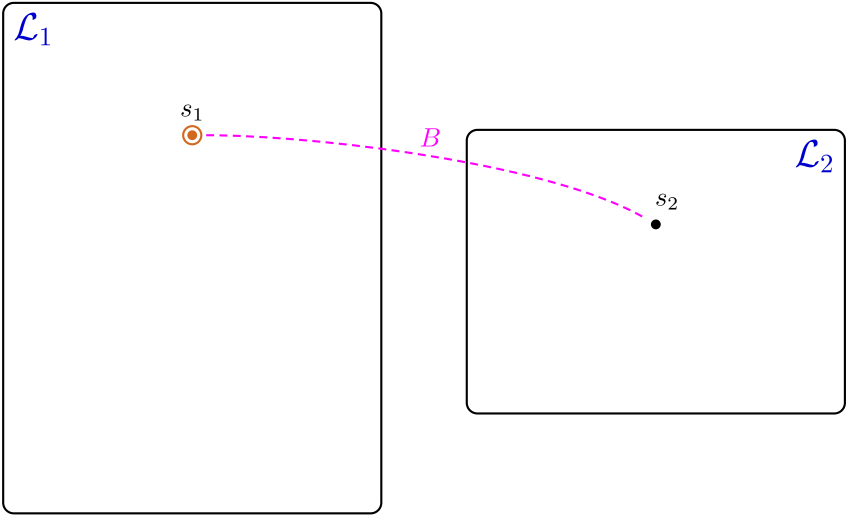
Bisimulation (back condition)



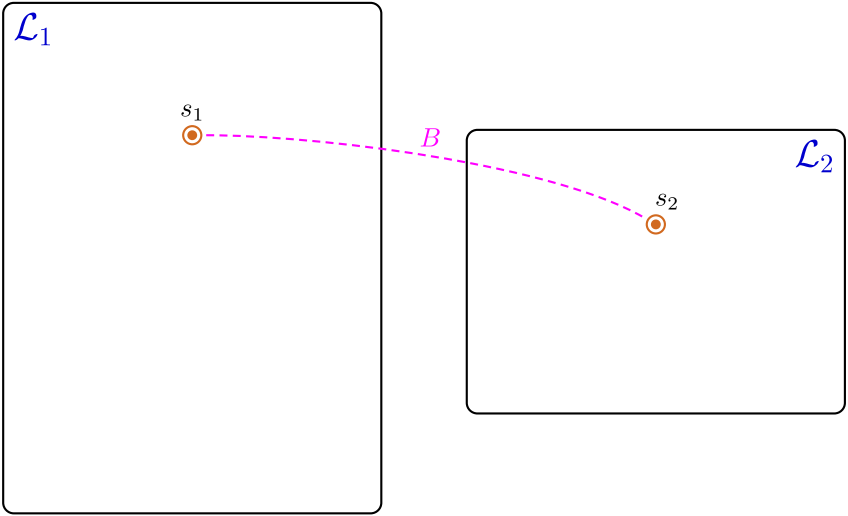
Bisimulation (back condition)



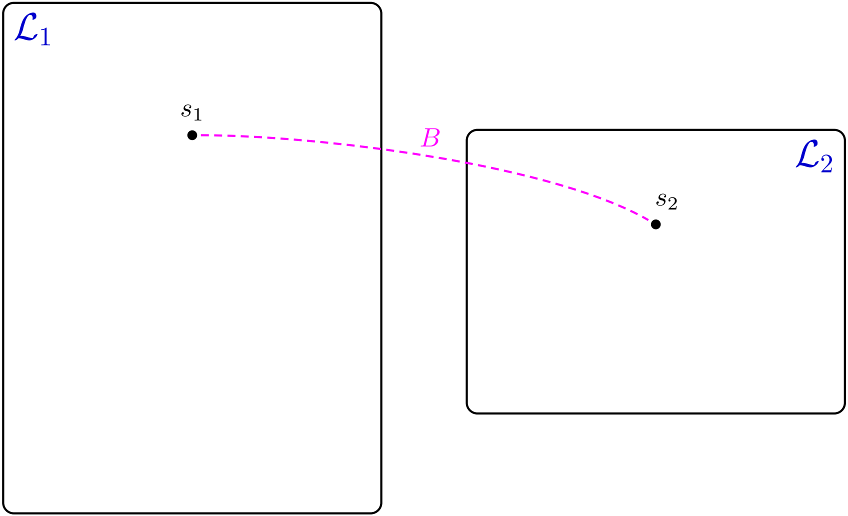
Bisimulation (termination condition)



Bisimulation (termination condition)



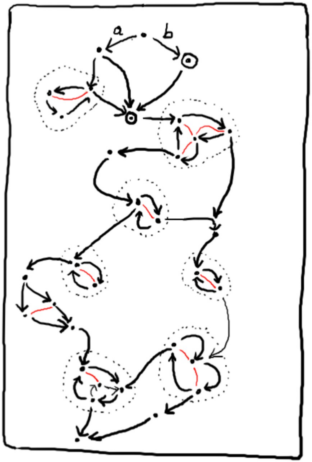
Bisimulation



Nearly collapsed LTSs

strongly connected components (scc's)

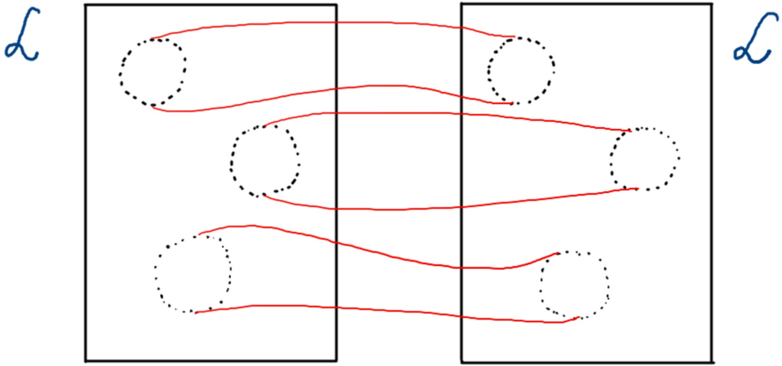
termination permitted



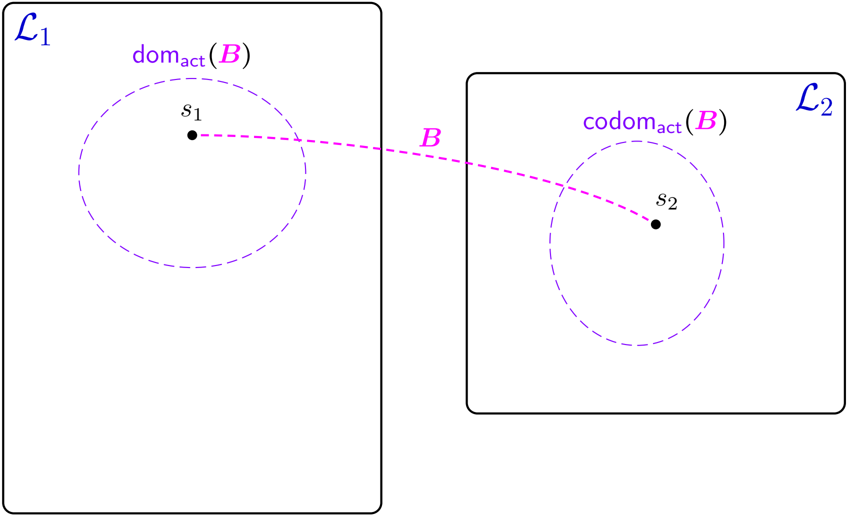
— bisimilarity

is contained within scc's

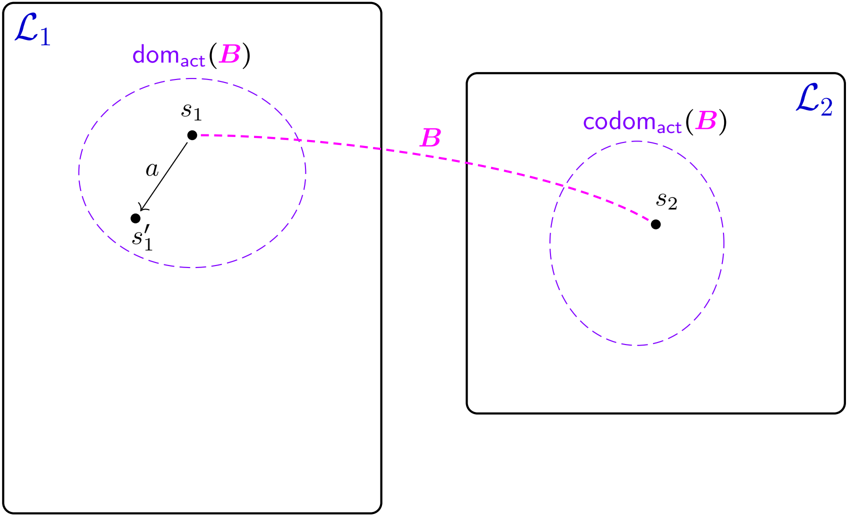
Bisimulation/bisimulating slices



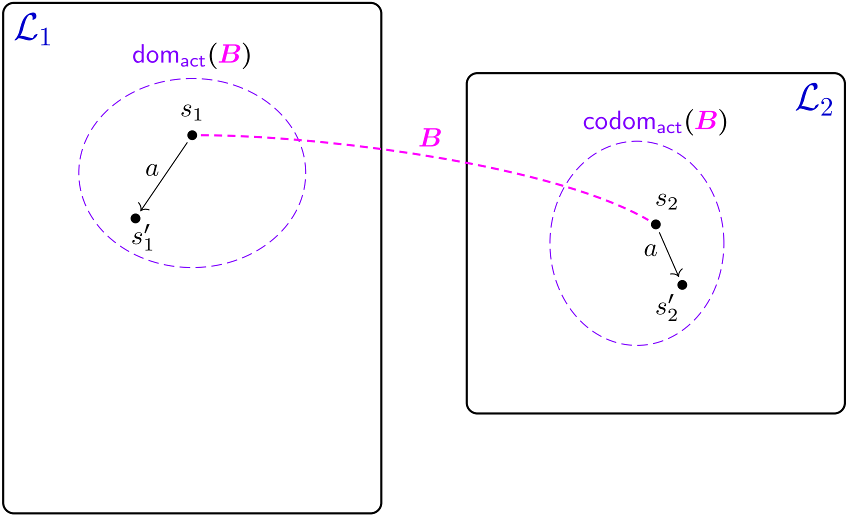
Bisimulating slice



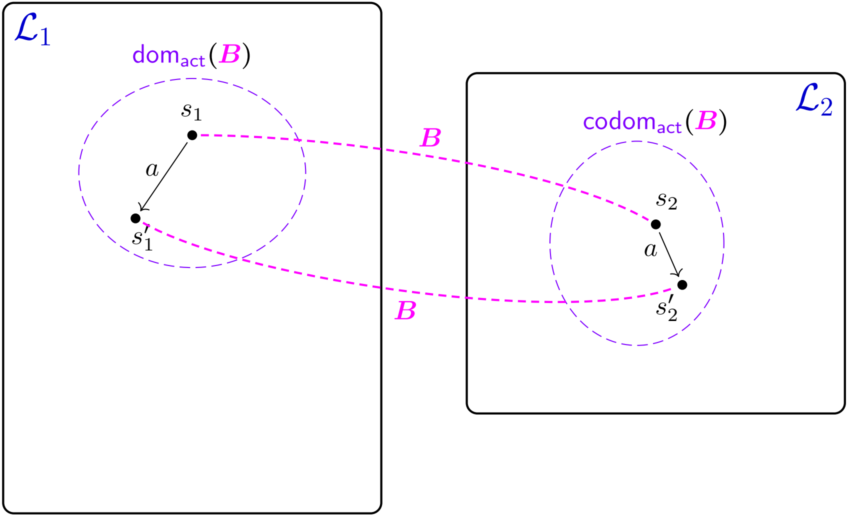
Bisimulating slice (forth condition)



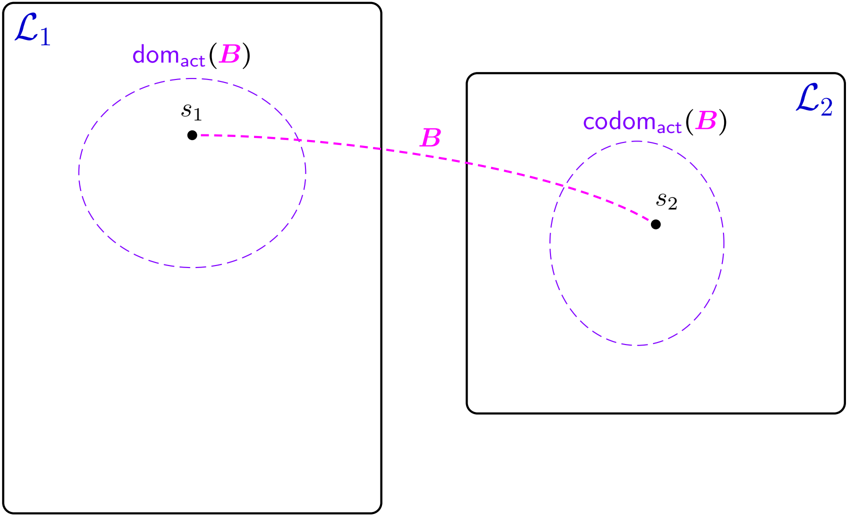
Bisimulating slice (forth condition)



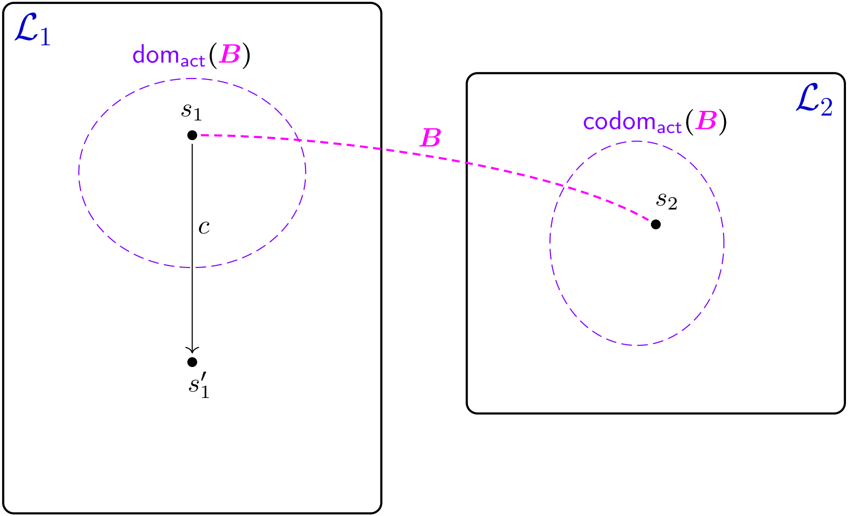
Bisimulating slice (forth condition)



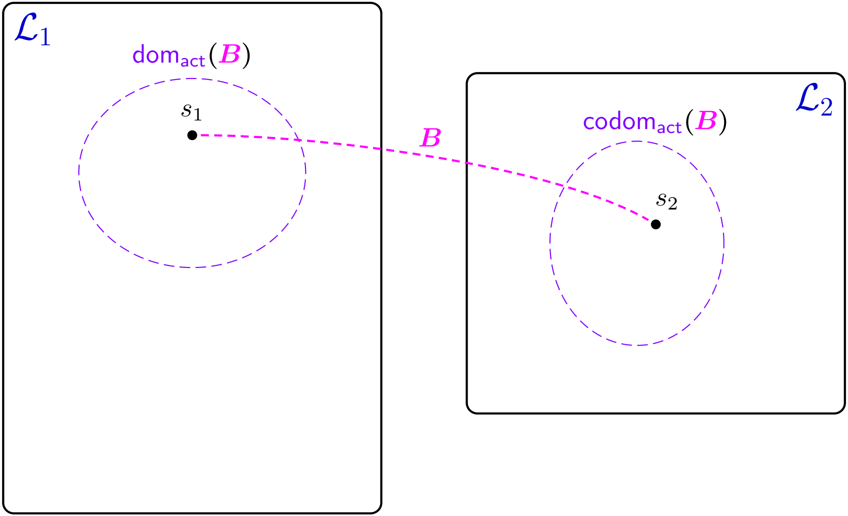
Bisimulating slice (forth condition)



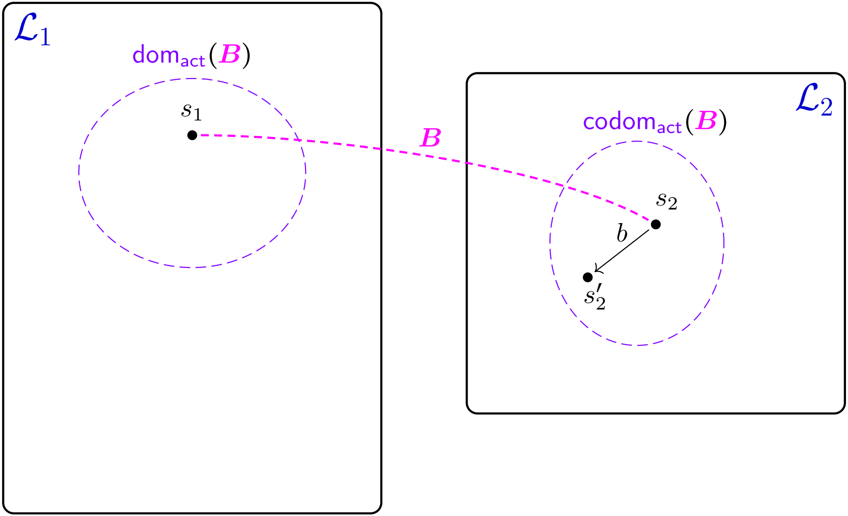
Bisimulating slice (forth condition)



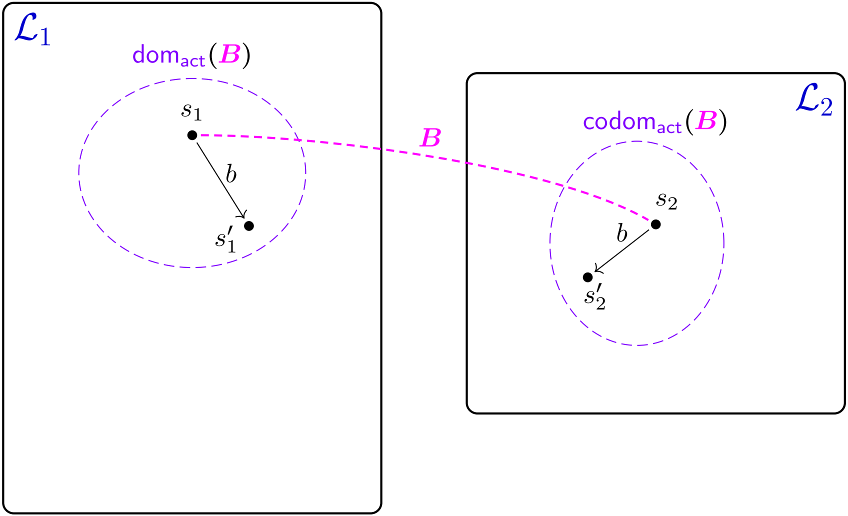
Bisimulating slice



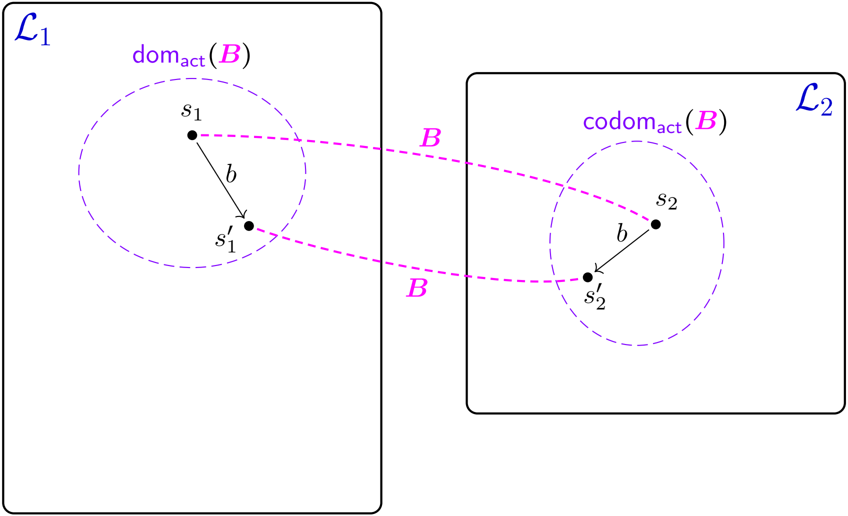
Bisimulating slice (back condition)



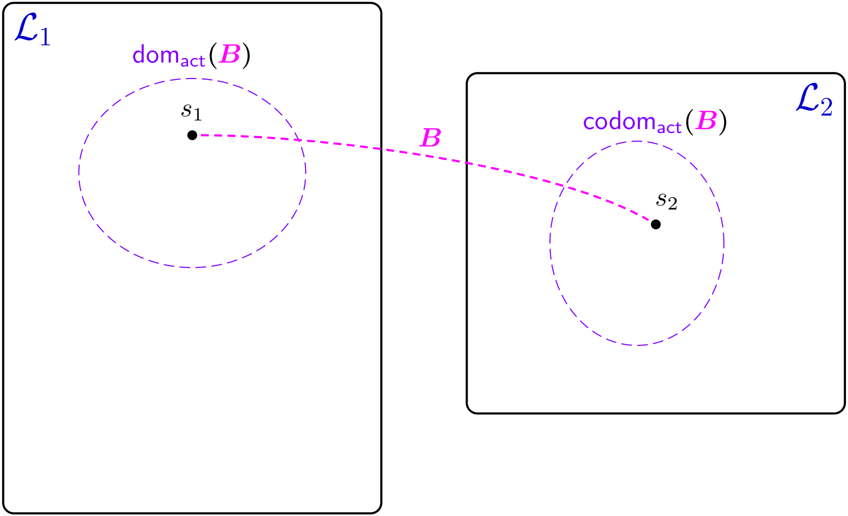
Bisimulating slice (back condition)



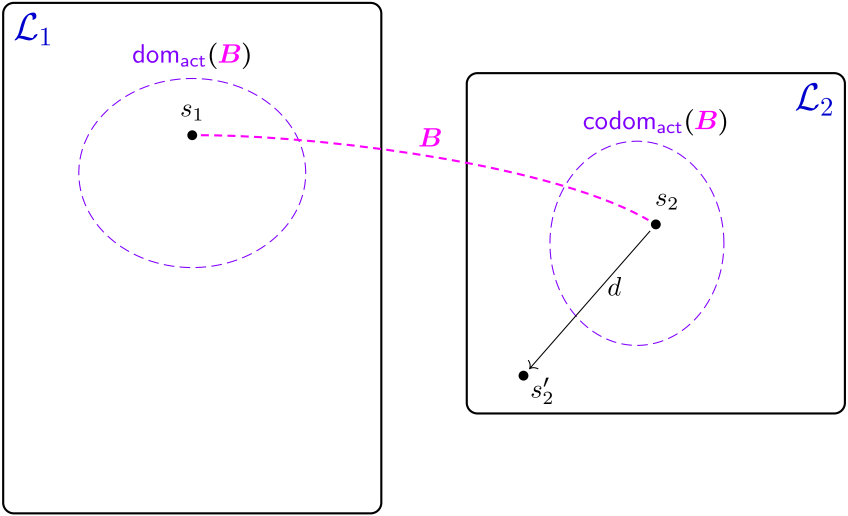
Bisimulating slice (back condition)



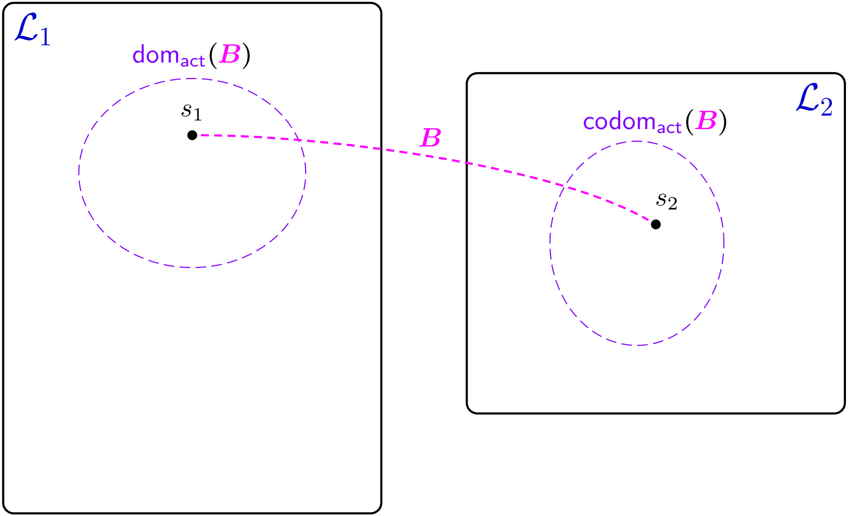
Bisimulating slice (back condition)



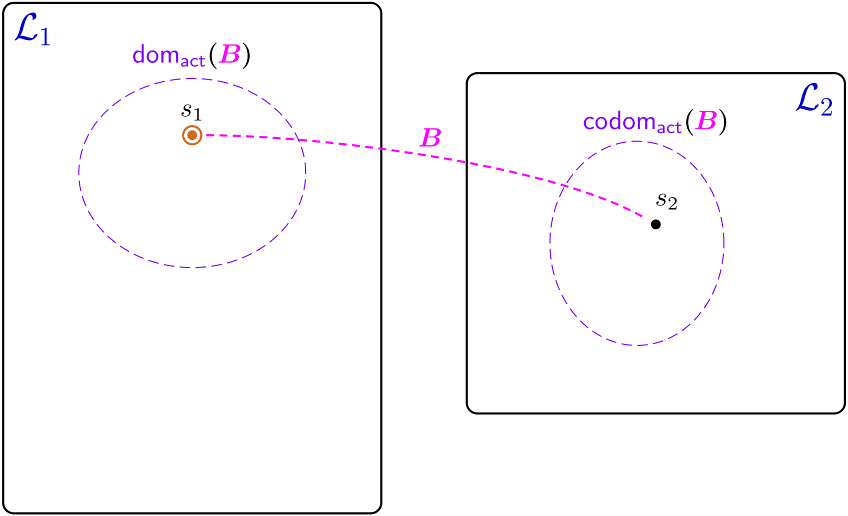
Bisimulating slice (back condition)



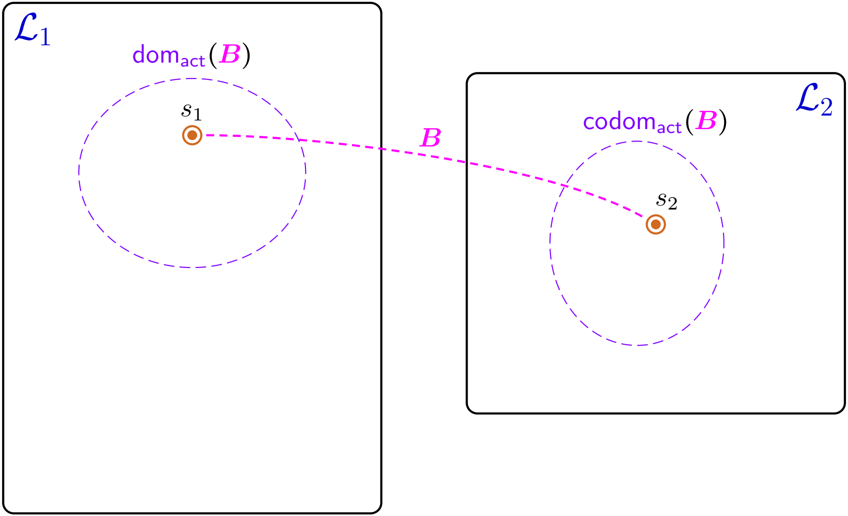
Bisimulating slice



Bisimulating slice (termination condition)



Bisimulating slice (termination condition)



Bisimulating/bisimulation slices: properties

Proposition

Bisimulating slice

$=$ *bisimulation on full sub-LTSs of active domain/codomain*

Bisimulating/bisimulation slices: properties

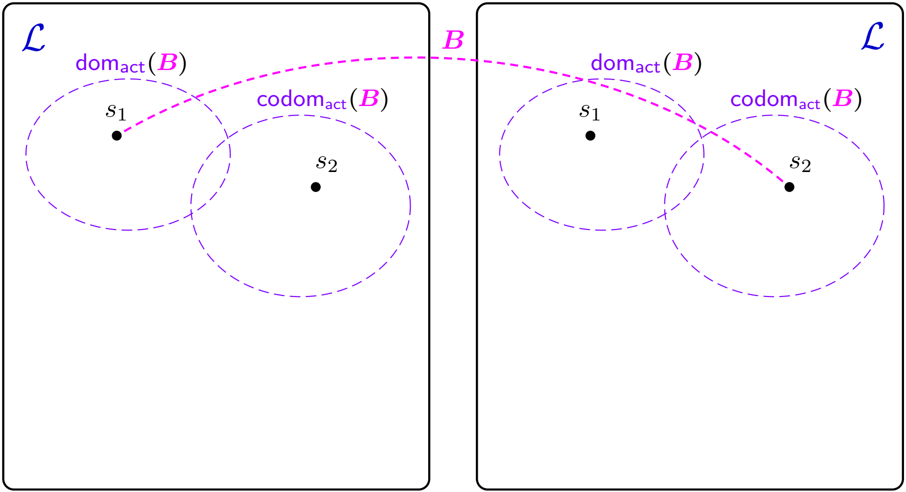
Proposition

Bisimulating slice
$=$ *bisimulation on full sub-LTSs of active domain/codomain*

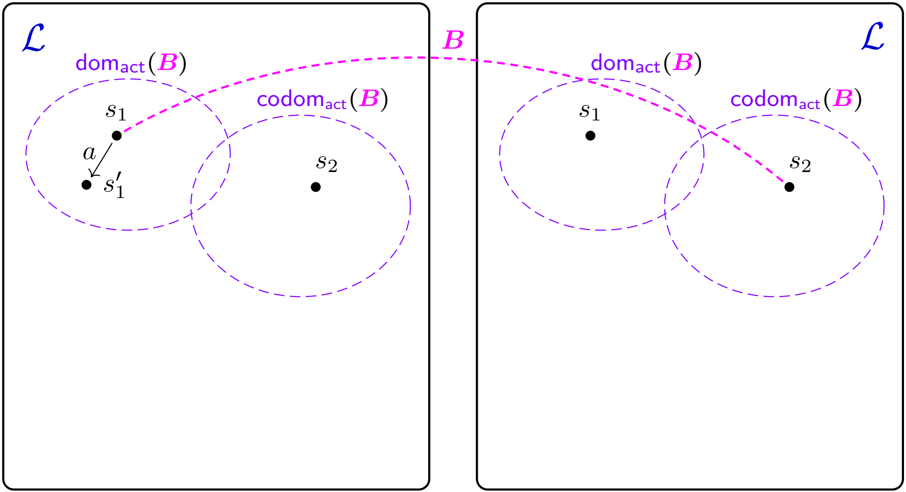
Proposition

Bisimulation
$=$ *bisimulating slice on transition-closed active domain/codomain*

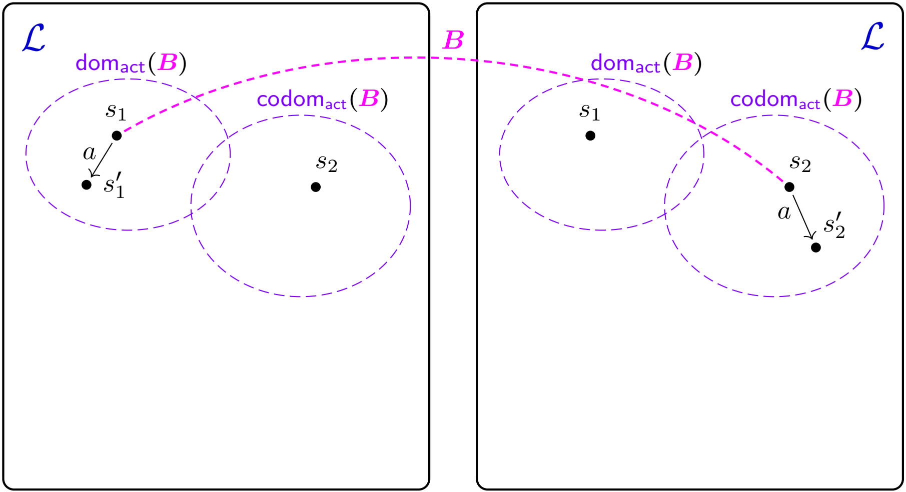
Grounded bisimulation slice



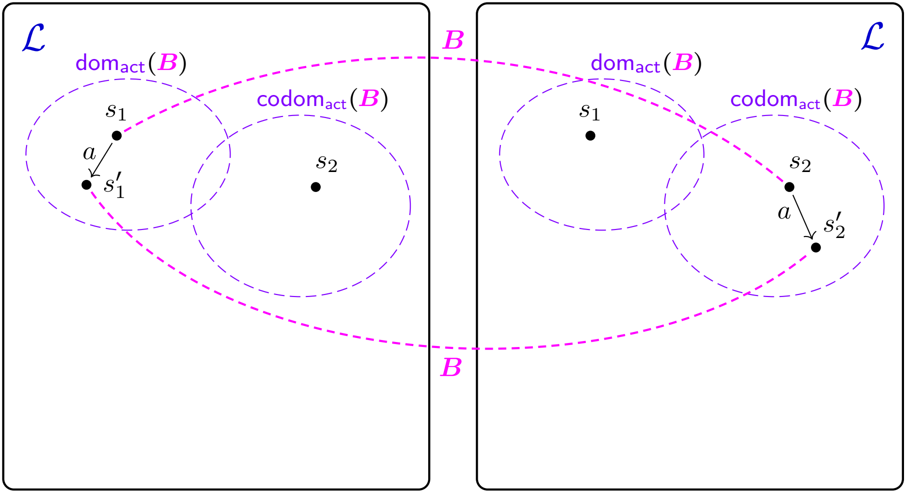
Grounded bisimulation slice (forth condition)



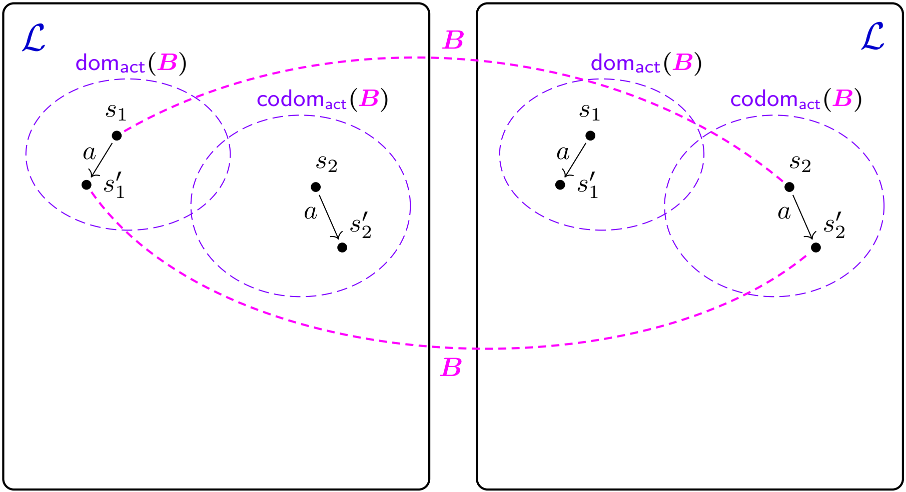
Grounded bisimulation slice (forth condition)



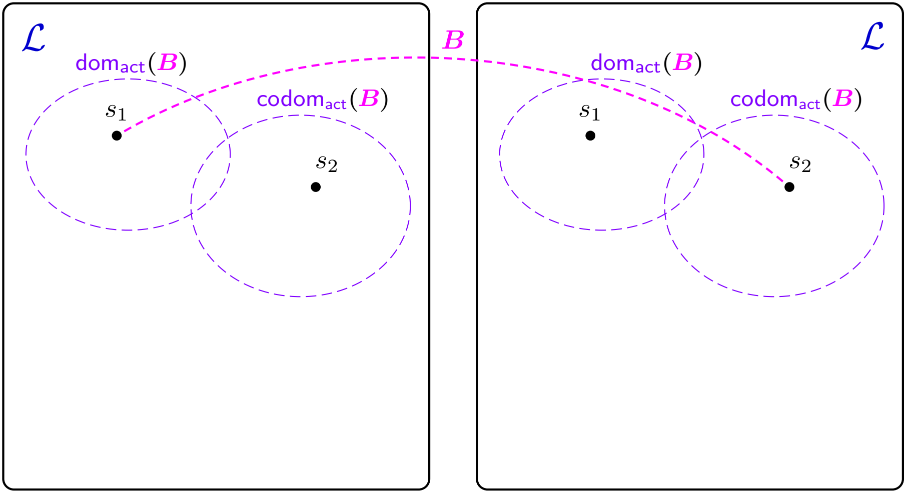
Grounded bisimulation slice (forth condition)



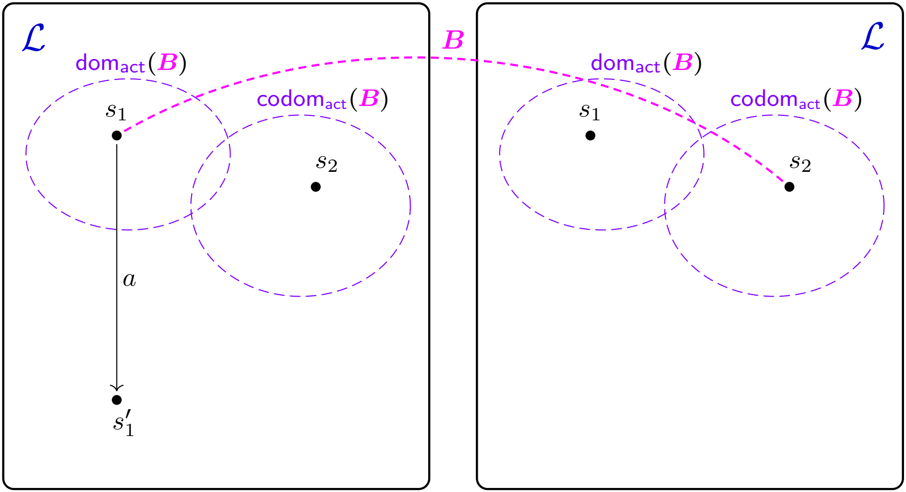
Grounded bisimulation slice (forth condition)



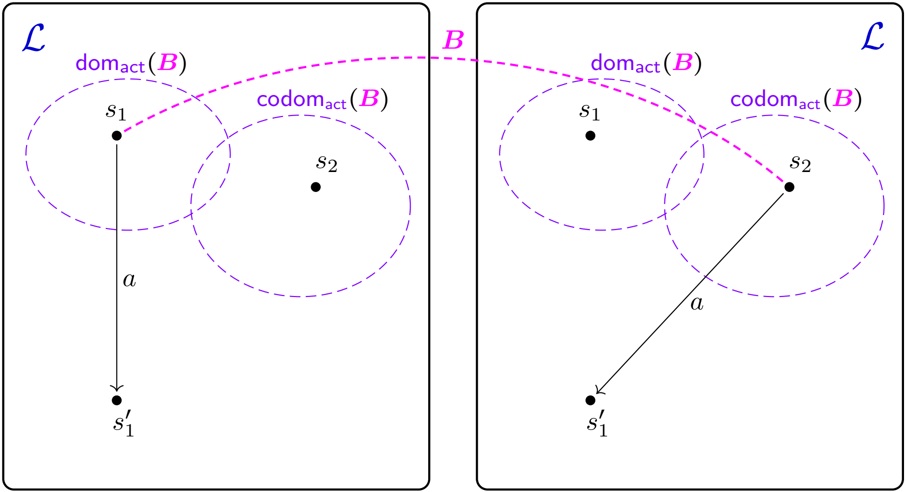
Grounded bisimulation slice (forth condition)



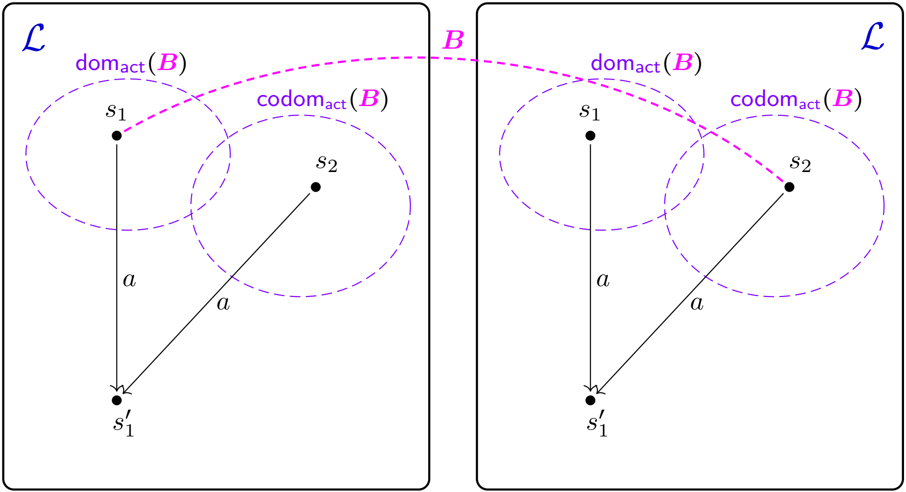
Grounded bisimulation slice (added forth condition)



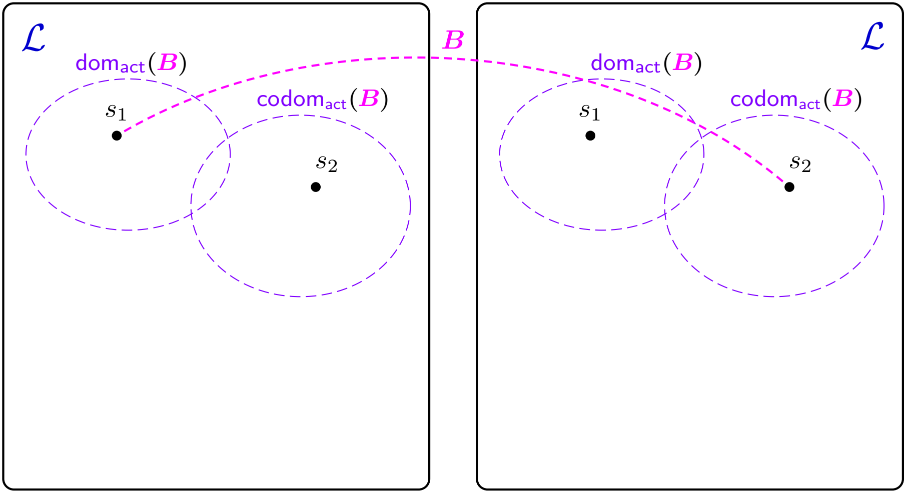
Grounded bisimulation slice (added forth condition)



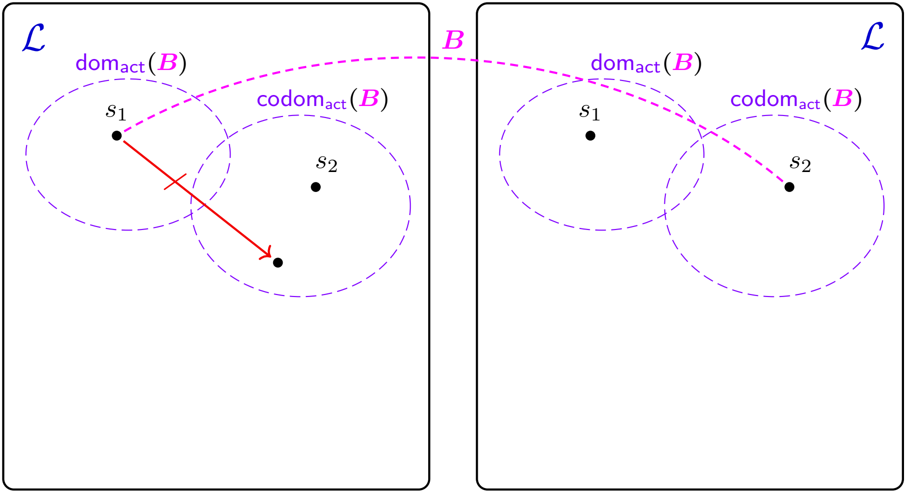
Grounded bisimulation slice (added forth condition)



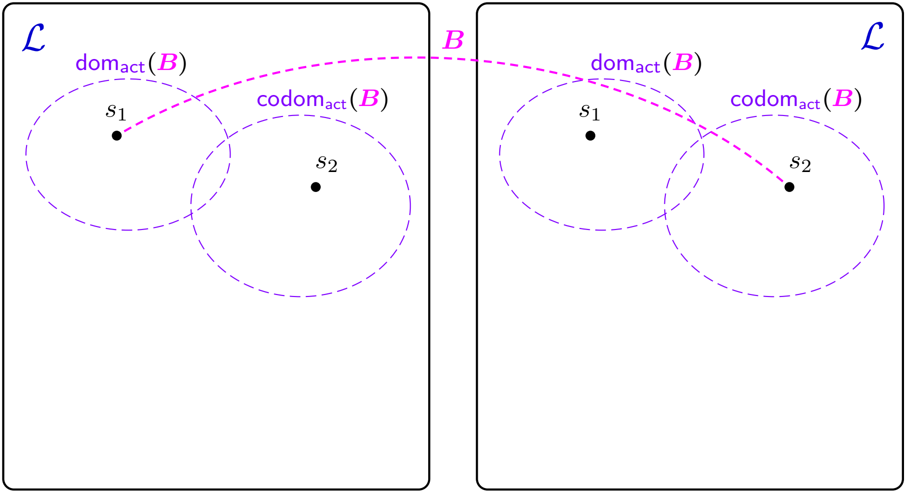
Grounded bisimulation slice (added forth condition)



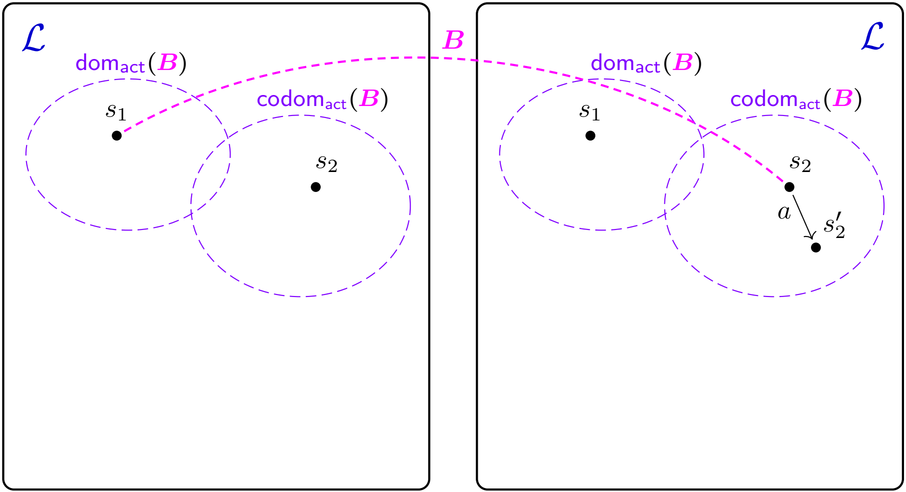
Grounded bisimulation slice (added forth condition)



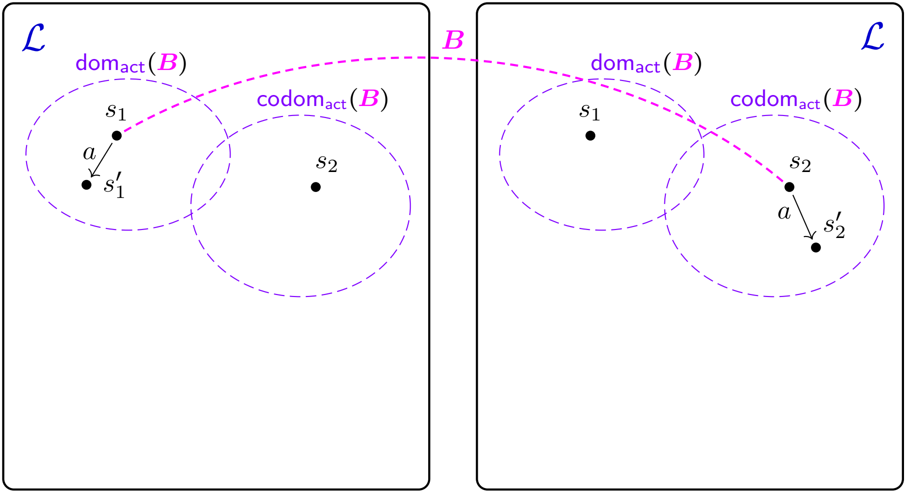
Grounded bisimulation slice



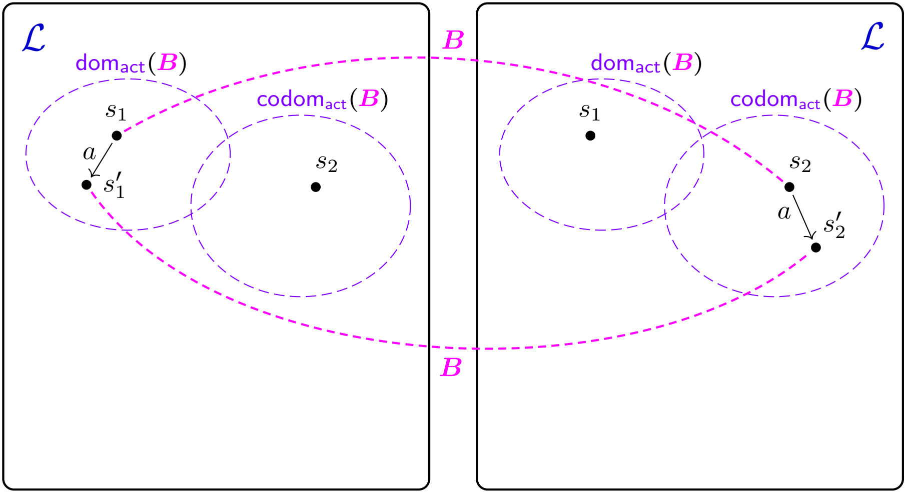
Grounded bisimulation slice (back condition)



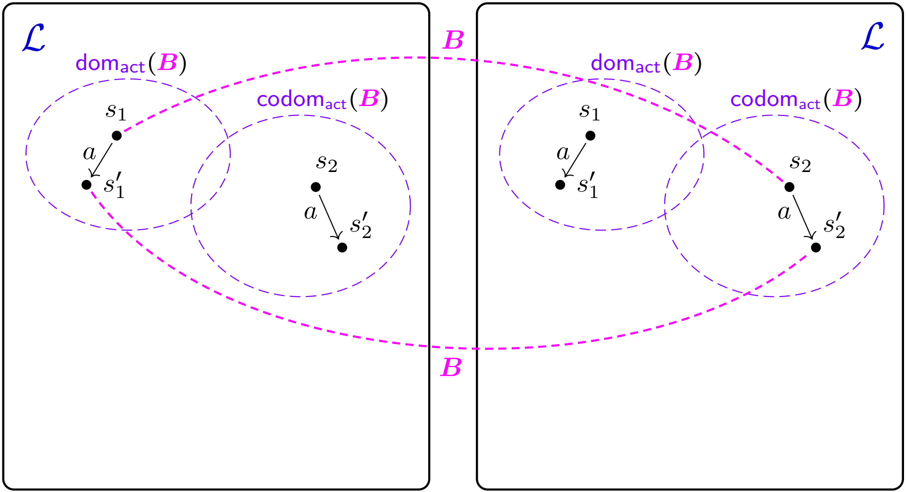
Grounded bisimulation slice (back condition)



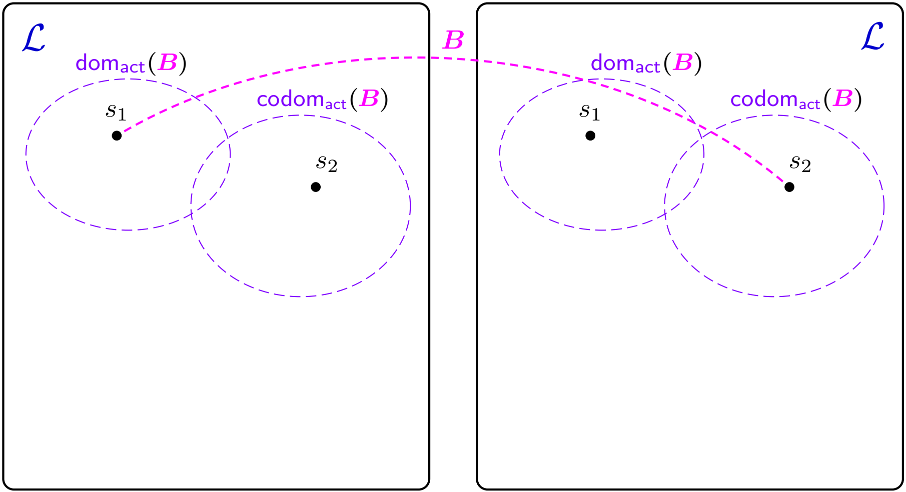
Grounded bisimulation slice (back condition)



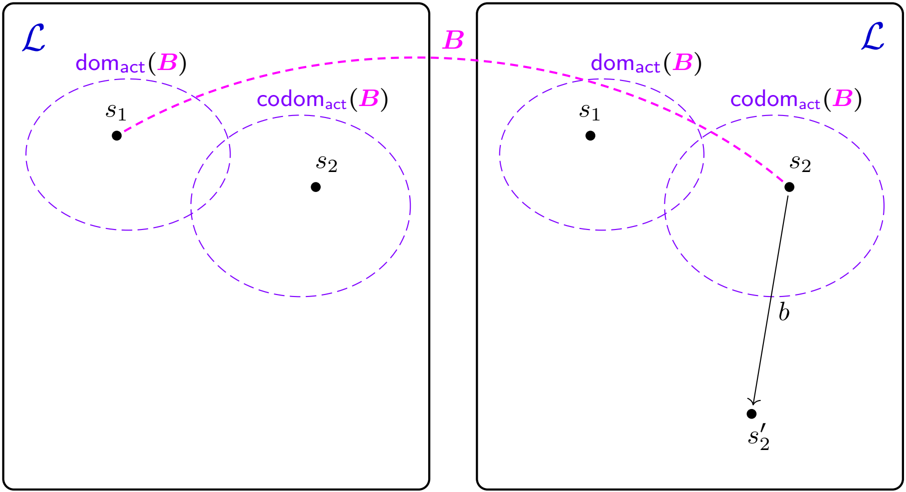
Grounded bisimulation slice (back condition)



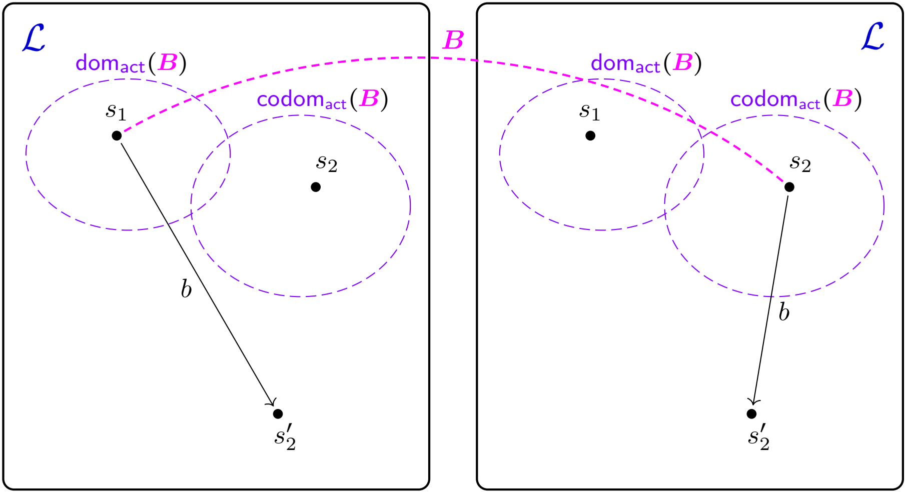
Grounded bisimulation slice (back condition)



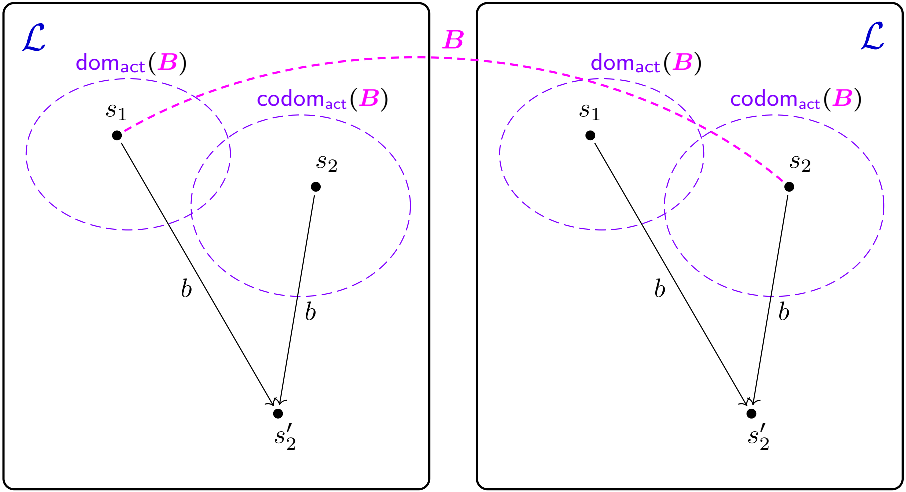
Grounded bisimulation slice (back condition)



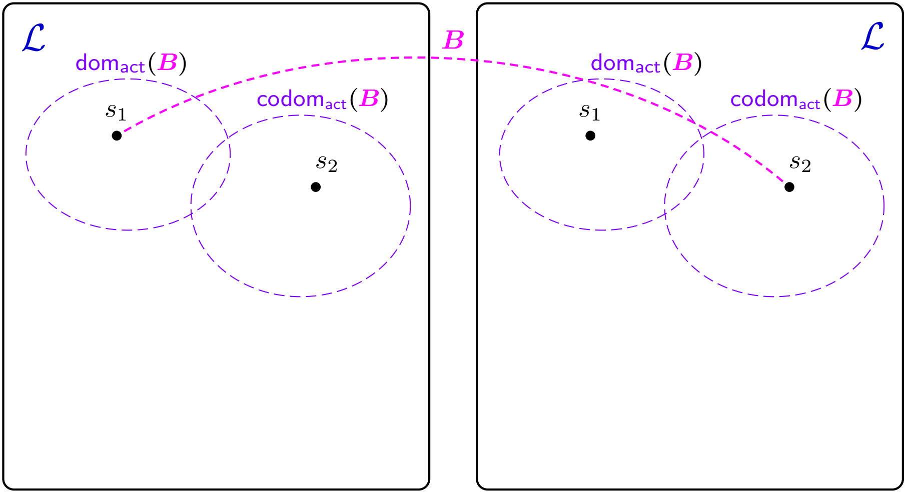
Grounded bisimulation slice (added back condition)



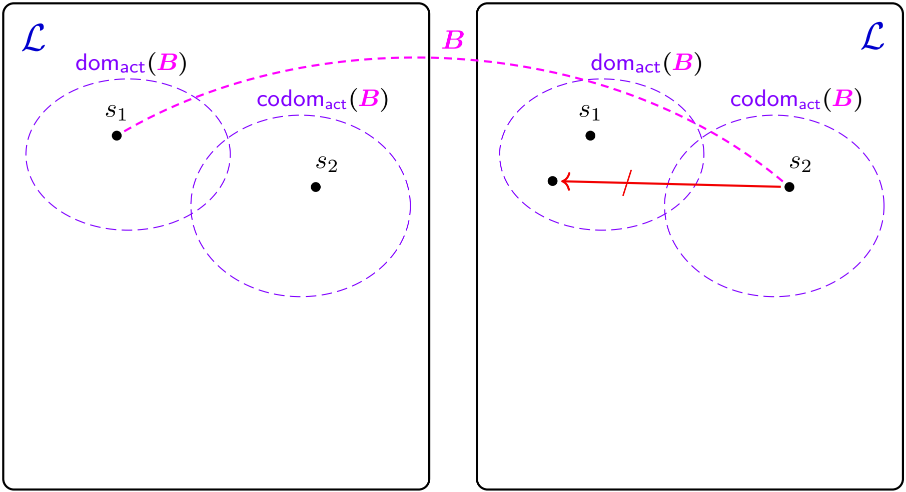
Grounded bisimulation slice (added back condition)



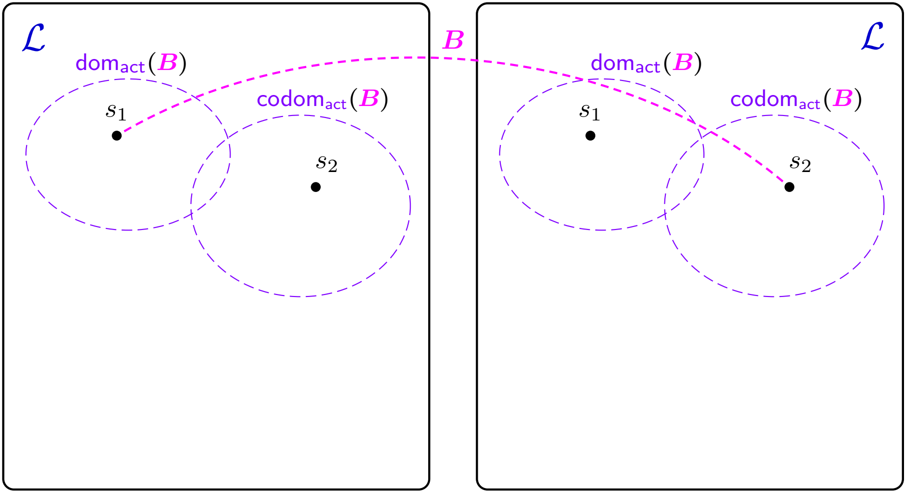
Grounded bisimulation slice (added back condition)



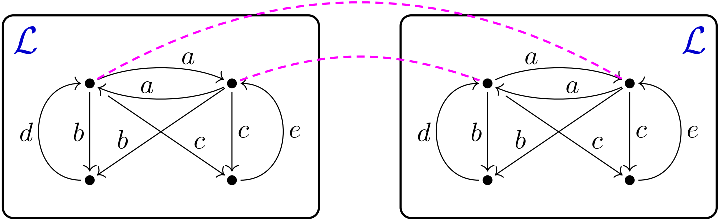
Grounded bisimulation slice (added back condition)



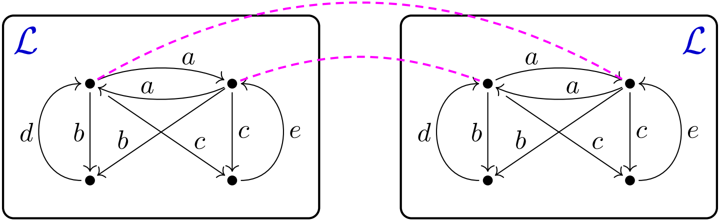
Grounded bisimulation slice



Grounded bisimulation slices: example



Grounded bisimulation slices: example



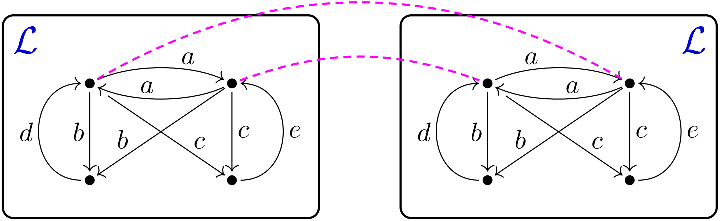
- B is grounded bisimulation slice,

Grounded bisimulation slices: example and property

Proposition (*Grounded bisimulation slices induce bisimulations*)

For every grounded bisimulation slice $\boldsymbol{B} \subseteq S \times S$ on LTS $\mathcal{L} = \langle S, A, \rightarrow, \downarrow \rangle$,

$$\overline{\overline{\boldsymbol{B}}} := \boldsymbol{B} \cup = \qquad \text{is a bisimulation on } \mathcal{L}.$$



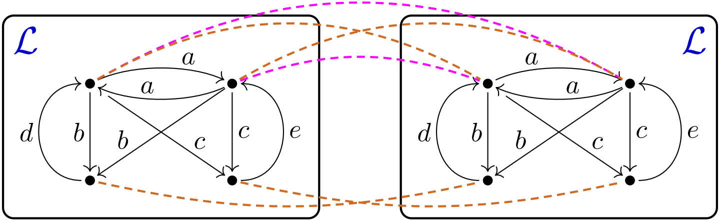
- $\boldsymbol{B}$ is grounded bisimulation slice,

Grounded bisimulation slices: example and property

Proposition (*Grounded bisimulation slices induce bisimulations*)

For every grounded bisimulation slice $\boldsymbol{B} \subseteq S \times S$ on LTS $\mathcal{L} = \langle S, A, \rightarrow, \downarrow \rangle$,

$$\overline{\overline{\boldsymbol{B}}} := \boldsymbol{B} \cup = \quad \text{is a bisimulation on } \mathcal{L}.$$



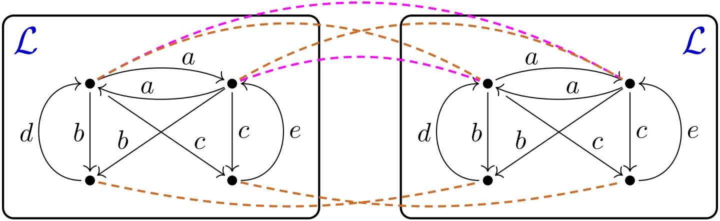
- $\boldsymbol{B}$ is grounded bisimulation slice,

Grounded bisimulation slices: example and property

Proposition (*Grounded bisimulation slices induce bisimulations*)

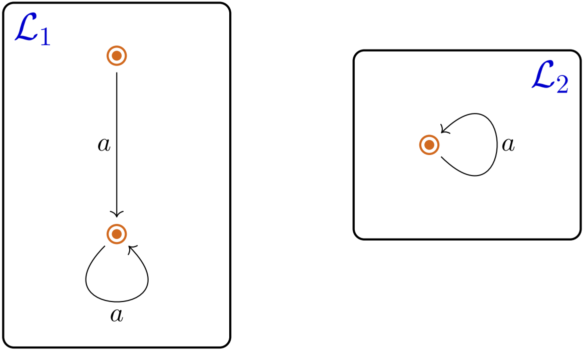
For every grounded bisimulation slice $B \subseteq S \times S$ on LTS $\mathcal{L} = \langle S, A, \to, \downarrow \rangle$,

$$\overline{\overline{B}} := B \cup = \quad \text{is a bisimulation on } \mathcal{L}.$$



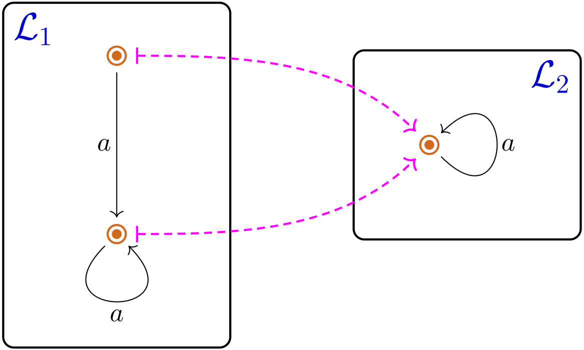
- B is grounded bisimulation slice,
- the bisimulation $\overline{\overline{B}}$ that extends B.

Motivation 2: specification transfer via funct. bisimulations



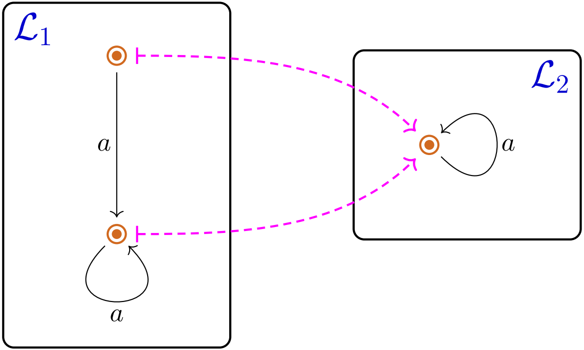
Motivation 2: specification transfer via funct. bisimulations

functional bisimulation



Motivation 2: specification transfer via funct. bisimulations

functional bisimulation

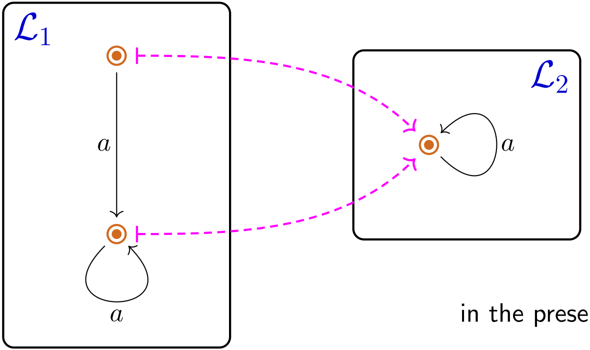


specification

a^*

Motivation 2: specification transfer via funct. bisimulations

functional bisimulation



specification

a^*

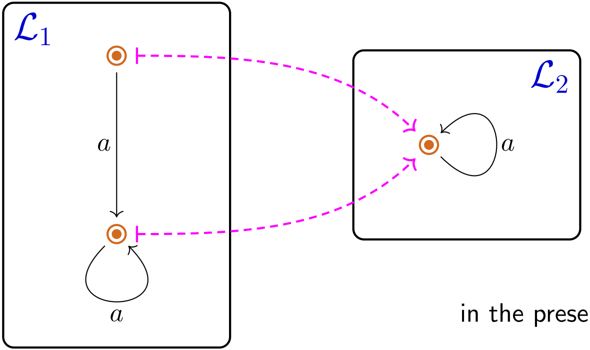
in the presence of axiom:

$$a^* = 1 + a \cdot a^*$$

Motivation 2: specification transfer via funct. bisimulations

pullback via

functional bisimulation



specification

a^*

in the presence of axiom:

$$a^* = 1 + a \cdot a^*$$

Motivation 2: specification transfer via funct. bisimulations

specifications

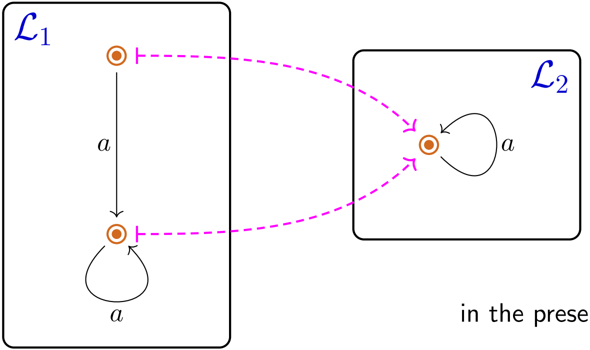
pullback via

functional bisimulation

specification

a^*

a^*



a^*

in the presence of axiom:

$$a^* = 1 + a \cdot a^*$$

Transfer functions, and local-transfer functions

Transfer function $\hat{=}$ functional bisimulation between LTSs

Definition

A *transfer (partial) function* between LTSs $\mathcal{L}_1$, $\mathcal{L}_2$ is:

- a partial function $\phi : S_1 \rightharpoonup S_2$
  whose graph $\{\langle v, \phi(v)\rangle \mid v \in S_1\}$ is a bisimulation betw. $\mathcal{L}_1$ and $\mathcal{L}_2$.

Transfer functions, and local-transfer functions

Transfer function $\hat{=}$ functional bisimulation between LTSs

Definition

A *transfer (partial) function* between LTSs $\mathcal{L}_1$, $\mathcal{L}_2$ is:

- a partial function $\phi : S_1 \rightharpoonup S_2$
  whose graph $\{\langle v, \phi(v)\rangle \mid v \in S_1\}$ is a bisimulation betw. $\mathcal{L}_1$ and $\mathcal{L}_2$.

Local-transfer function $\hat{=}$ functional grounded bisimulation slice on an LTS

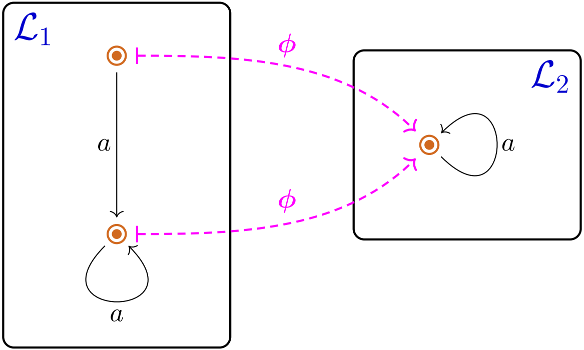
Definition

A *local-transfer function* on an LTS $\mathcal{L}$ is:

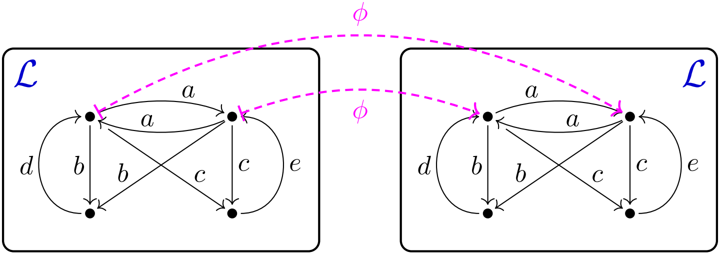
- a partial function $\phi : S_1 \rightharpoonup S_2$
  whose graph $\{\langle v, \phi(v)\rangle \mid v \in S\}$ is a grounded bisimulation slice on $\mathcal{L}$.

Transfer function (example)

ϕ is transfer function

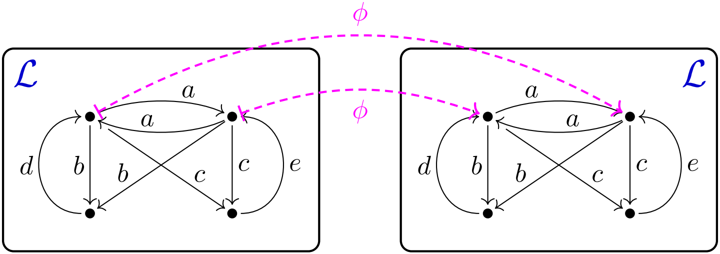


Local transfer function (example)



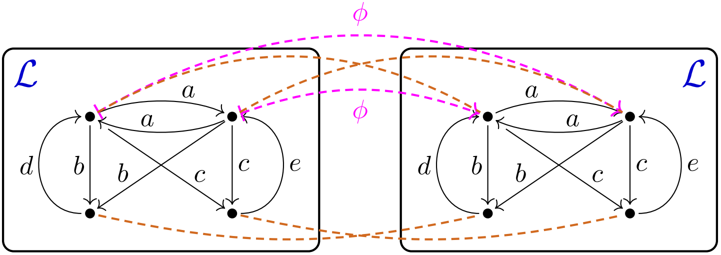
- ϕ is local transfer function on $\mathcal{L}$,

Local transfer function (example)



- ϕ is local transfer function on $\mathcal{L}$,
- but ϕ cannot be extended into a transfer function on $\mathcal{L}$.

Local transfer function (example)



- ϕ is local transfer function on $\mathcal{L}$,
- but ϕ cannot be extended into a transfer function on $\mathcal{L}$.

Goal: Lifting a local transfer function

$$\mathcal{L} \xrightarrow{\quad\phi\quad} \mathcal{L}$$

local transfer function ϕ

Goal: Lifting a local transfer function to a transfer function

$$\mathcal{L} \xrightarrow{\quad\phi\quad} \mathcal{L}$$

local transfer function ϕ

$$\mathsf{E}_W(\mathcal{L}) \xrightarrow{\quad\widehat{\phi}\quad} \mathsf{E}_W(\mathcal{L})$$

transfer function $\widehat{\phi}$

elevation of W above $\mathcal{L}$

for $W = \mathrm{dom}(\phi) \cup \mathrm{ran}(\phi)$

Goal: Lifting a local transfer function to a transfer function

$$\begin{tikzcd}
\mathcal{L} \arrow[r, "\phi"] & \mathcal{L} \\
\mathsf{E}_{W}(\mathcal{L}) \arrow[u, "\pi_1"] \arrow[r, "\widehat{\phi}"] & \mathsf{E}_{W}(\mathcal{L}) \arrow[u, "\pi_1"']
\end{tikzcd}$$

local transfer function ϕ

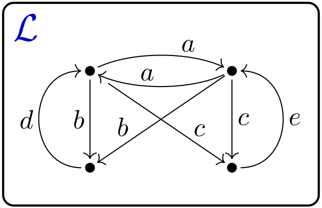
projections π_1

transfer function $\widehat{\phi}$

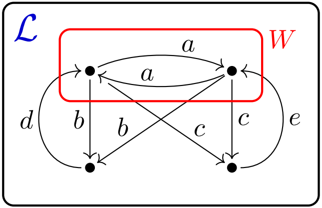
elevation of W above $\mathcal{L}$

for $W = \mathrm{dom}(\phi) \cup \mathrm{ran}(\phi)$

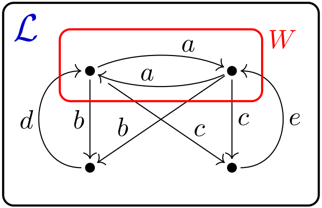
Elevation of a vertex set above LTS (example)

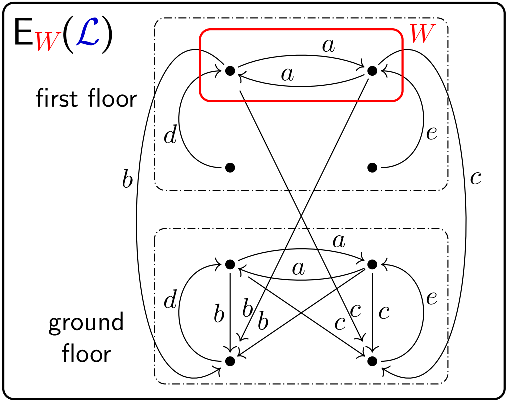


Elevation of a vertex set above LTS (example)



Elevation of a vertex set above LTS (example)





Local-transfer function ⇒ transfer function on elevation

Proposition

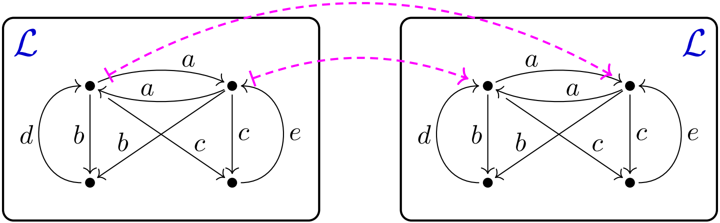
$$
\begin{tikzcd}
\mathcal{L} \arrow[r, "\phi"] & \mathcal{L} \\
\mathsf{E}_{W}(\mathcal{L}) \arrow[u, "\pi_1"] \arrow[r, "\widehat{\phi}"] & \mathsf{E}_{W}(\mathcal{L}) \arrow[u, "\pi_1"']
\end{tikzcd}
$$

local transfer function ϕ

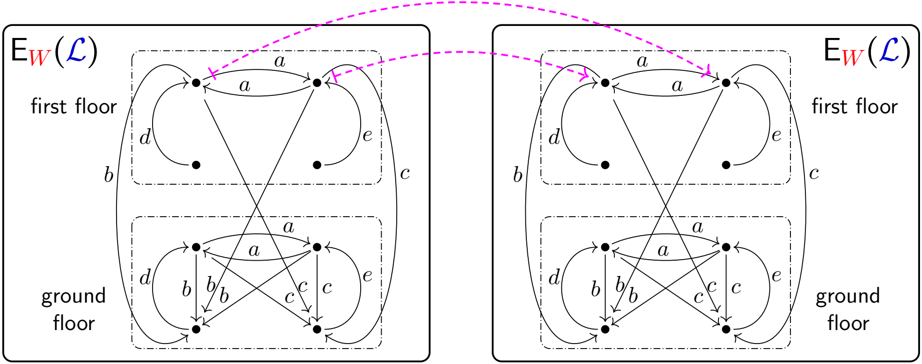
projections π_1
are transfer functions

transfer function $\widehat{\phi}$

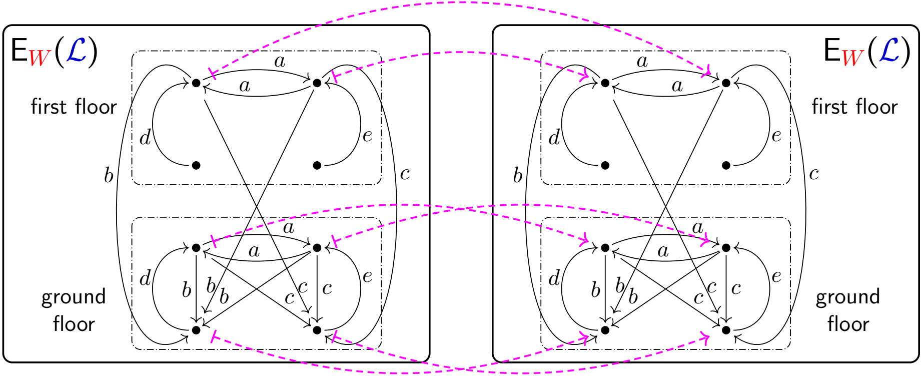
Local-transfer function ⇒ transfer function on elevation



Local-transfer function ⇒ transfer function on elevation



Local-transfer function ⇒ transfer function on elevation



Local-transfer function ⇒ transfer function on elevation

Proposition

Let $\mathcal{L} = \langle S, A, \to, \downarrow \rangle$ be an LTS.
Every local-transfer function $\phi : S \rightharpoonup S$ on $\mathcal{L}$ lifts to a transfer function on the elevation $\mathsf{E}_{\mathsf{field}(\phi)}(\mathcal{L})$ of $W := \mathsf{field}(\phi) = \mathsf{dom}(\phi) \cup \mathsf{ran}(\phi)$ above $\mathcal{L}$

where: $\widehat{\phi} : (S \times \{\mathbf{0}, \mathbf{1}\}) \longrightarrow (S \times \{\mathbf{0}, \mathbf{1}\})$

$$\langle s, \boldsymbol{i} \rangle \longmapsto \widehat{\phi}(\langle s, \boldsymbol{i} \rangle) := \begin{cases} \langle \phi(t), \boldsymbol{i} \rangle & \text{if } \boldsymbol{i} = \mathbf{1} \wedge t \in \mathsf{dom}(\phi) , \\ \text{undefined} & \text{if } \boldsymbol{i} = \mathbf{1} \wedge t \notin \mathsf{dom}(\phi) , \\ \langle t, \boldsymbol{i} \rangle & \text{if } \boldsymbol{i} = \mathbf{0} . \end{cases}$$

Local-transfer function ⇒ transfer function on elevation

Proposition

Let $\mathcal{L} = \langle S, A, \to, \downarrow \rangle$ be an LTS.
Every local-transfer function $\phi : S \rightharpoonup S$ on $\mathcal{L}$ lifts to a transfer function on the elevation $\mathsf{E}_{\mathsf{field}(\phi)}(\mathcal{L})$ of $W := \mathsf{field}(\phi) = \mathsf{dom}(\phi) \cup \mathsf{ran}(\phi)$ above $\mathcal{L}$ such that the following diagram commutes $\boxed{\text{on the first floor } (\mathbf{1})}$:

$$\begin{tikzcd}
\mathcal{L} \arrow[rr, "\phi"] & & \mathcal{L} \\
\mathsf{E}_W(\mathcal{L})^{\boxed{(\mathbf{1})}} \arrow[u, "\pi_1"] \arrow[rr, "\widehat{\phi}"] & & \mathsf{E}_W(\mathcal{L})^{\boxed{(\mathbf{1})}} \arrow[u, "\pi_1"']
\end{tikzcd}$$

local transfer function ϕ
projections π_1
are transfer functions
transfer function $\widehat{\phi}$

where: $\widehat{\phi} : (S \times \{\mathbf{0}, \mathbf{1}\}) \longrightarrow (S \times \{\mathbf{0}, \mathbf{1}\})$

$$\langle s, \boldsymbol{i} \rangle \longmapsto \widehat{\phi}(\langle s, \boldsymbol{i} \rangle) := \begin{cases} \langle \phi(t), \boldsymbol{i} \rangle & \text{if } \boldsymbol{i} = \mathbf{1} \wedge t \in \mathsf{dom}(\phi) \ , \\ \text{undefined} & \text{if } \boldsymbol{i} = \mathbf{1} \wedge t \notin \mathsf{dom}(\phi) \ , \\ \langle t, \boldsymbol{i} \rangle & \text{if } \boldsymbol{i} = \mathbf{0} \ . \end{cases}$$

Local-transfer function ⇒ transfer function on elevation

Proposition

Let $\mathcal{L} = \langle S, A, \to, \downarrow\rangle$ be an LTS.
Every local-transfer function $\phi : S \rightharpoonup S$ on $\mathcal{L}$ lifts to a transfer function on the elevation $\mathsf{E}_{\mathsf{field}(\phi)}(\mathcal{L})$ of $W := \mathsf{field}(\phi) = \mathsf{dom}(\phi) \cup \mathsf{ran}(\phi)$ above $\mathcal{L}$ such that the following diagram commutes $\boxed{\text{on the ground floor } (\mathbf{0})}$:

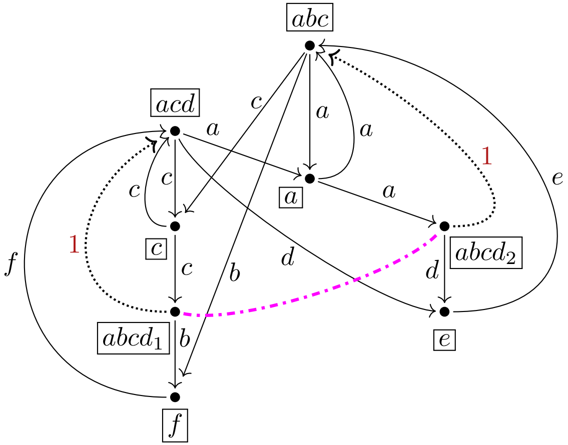
$$\begin{tikzcd}
\mathcal{L} \arrow[r, "\mathsf{id}_V"] & \mathcal{L} \\
\mathsf{E}_W(\mathcal{L})^{\boxed{(\mathbf{0})}} \arrow[u, "\pi_1"] \arrow[r, "\widehat{\phi}"] & \mathsf{E}_W(\mathcal{L})^{\boxed{(\mathbf{0})}} \arrow[u, "\pi_1"']
\end{tikzcd}$$

local transfer function ϕ
projections π_1
are transfer functions
transfer function $\widehat{\phi}$

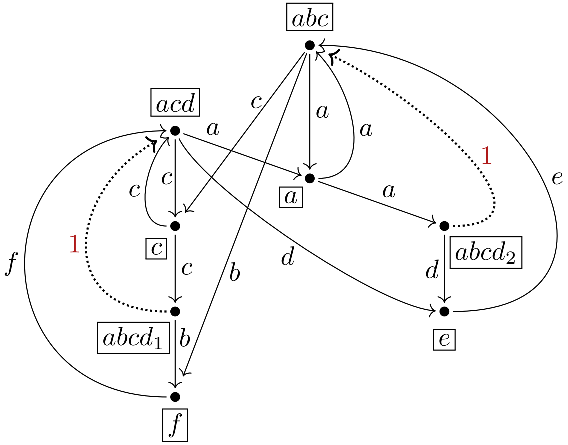
where: $\widehat{\phi} : (S \times \{\mathbf{0}, \mathbf{1}\}) \longrightarrow (S \times \{\mathbf{0}, \mathbf{1}\})$

$$\langle s, \boldsymbol{i}\rangle \longmapsto \widehat{\phi}(\langle s, \boldsymbol{i}\rangle) := \begin{cases} \langle \phi(t), \boldsymbol{i}\rangle & \text{if } \boldsymbol{i} = \mathbf{1} \wedge t \in \mathsf{dom}(\phi)\ , \\ \text{undefined} & \text{if } \boldsymbol{i} = \mathbf{1} \wedge t \notin \mathsf{dom}(\phi)\ , \\ \langle t, \boldsymbol{i}\rangle & \text{if } \boldsymbol{i} = \mathbf{0}\ . \end{cases}$$

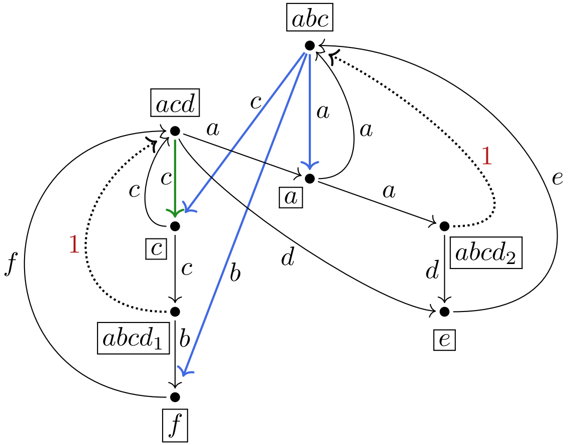
Application aimed at: nearly collapsed 1-LTSs



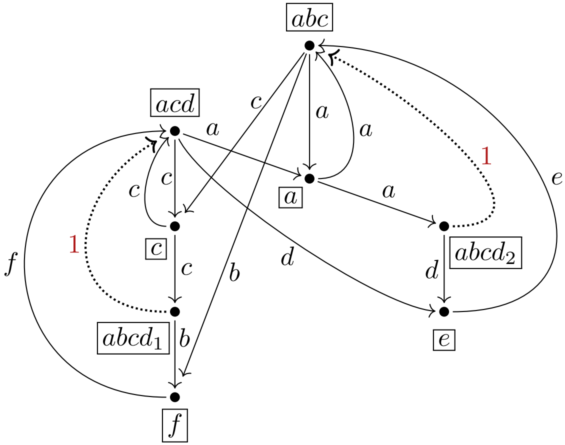
Application aimed at: nearly collapsed 1-LTSs



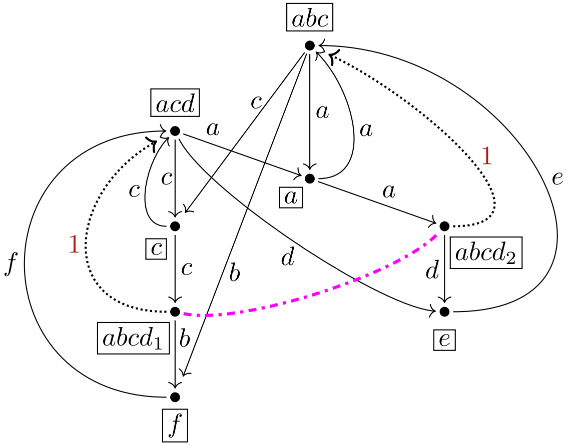
Application aimed at: nearly collapsed 1-LTSs



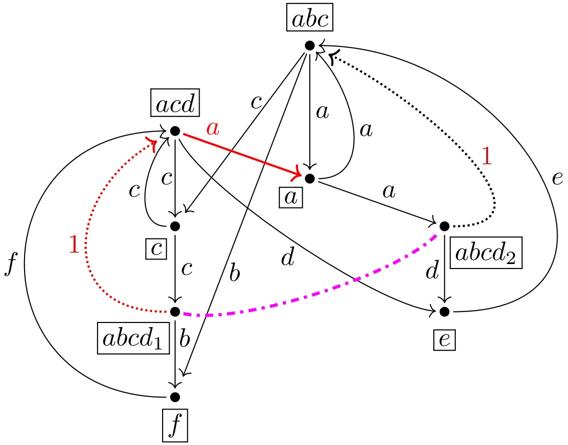
Application aimed at: nearly collapsed 1-LTSs



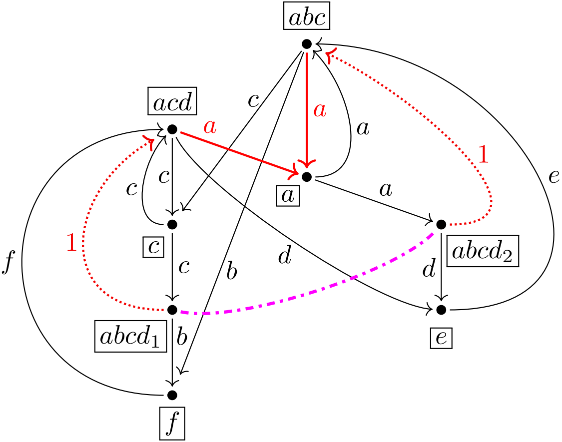
Application aimed at: nearly collapsed 1-LTSs



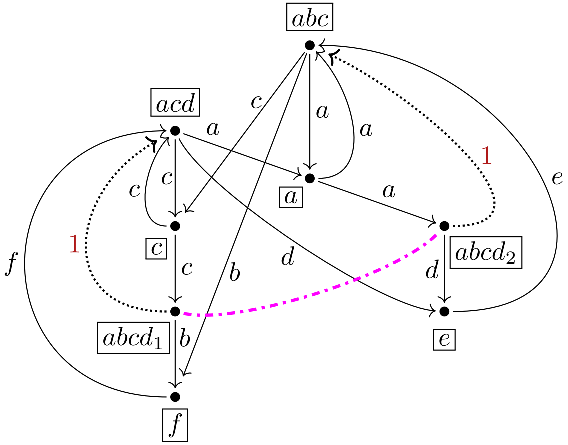
Application aimed at: nearly collapsed 1-LTSs



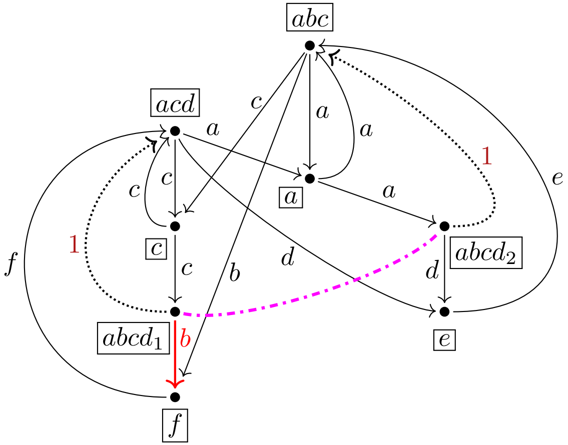
Application aimed at: nearly collapsed 1-LTSs



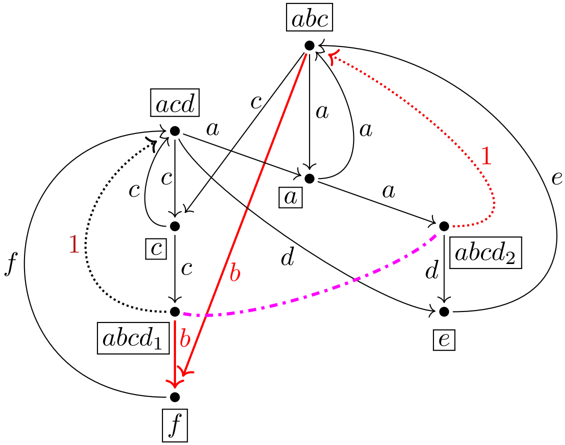
Application aimed at: nearly collapsed 1-LTSs



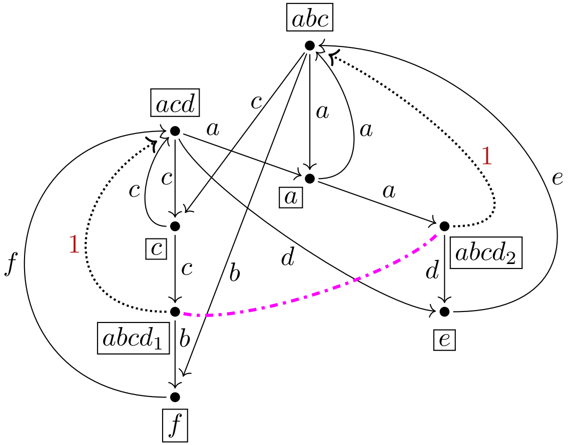
Application aimed at: nearly collapsed 1-LTSs



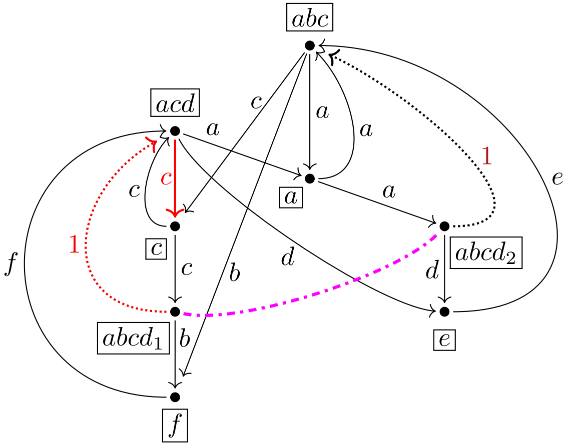
Application aimed at: nearly collapsed 1-LTSs



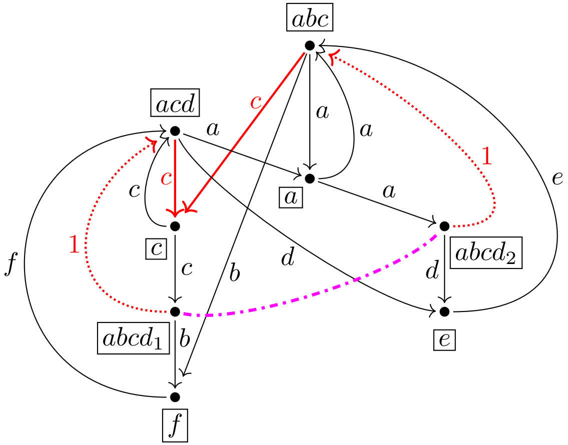
Application aimed at: nearly collapsed 1-LTSs



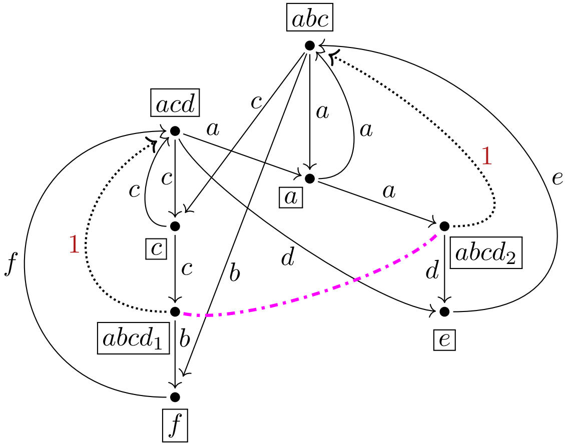
Application aimed at: nearly collapsed 1-LTSs



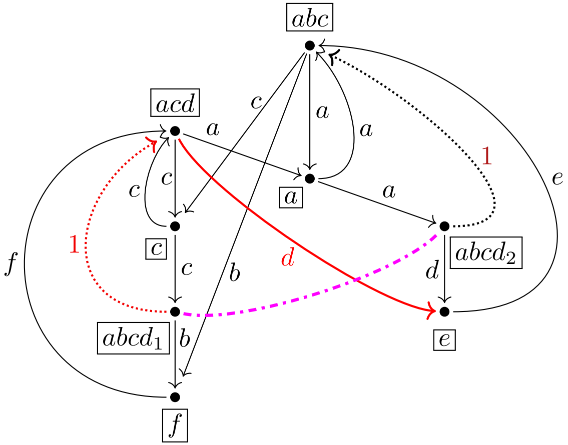
Application aimed at: nearly collapsed 1-LTSs



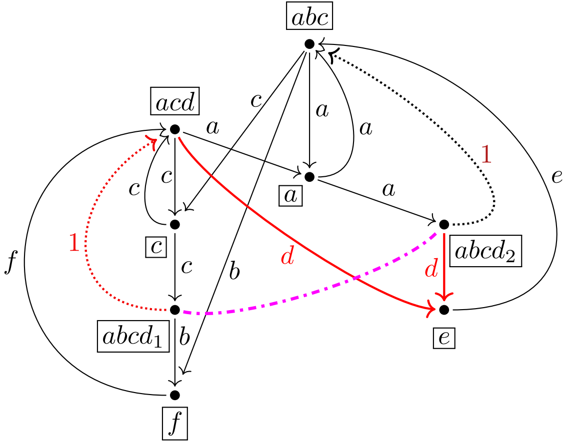
Application aimed at: nearly collapsed 1-LTSs



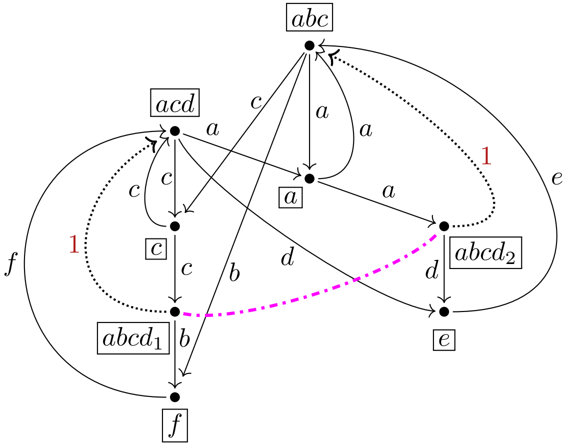
Application aimed at: nearly collapsed 1-LTSs



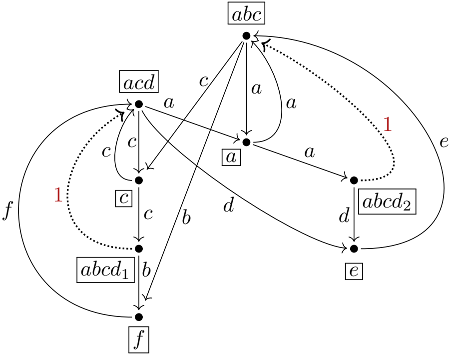
Application aimed at: nearly collapsed 1-LTSs

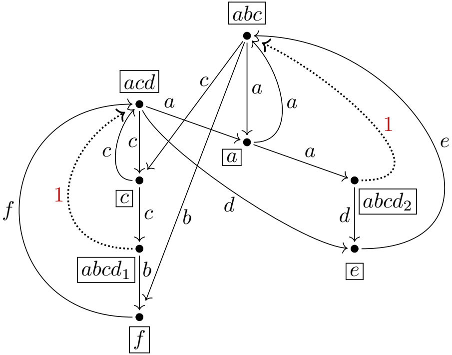


Application aimed at: nearly collapsed 1-LTSs

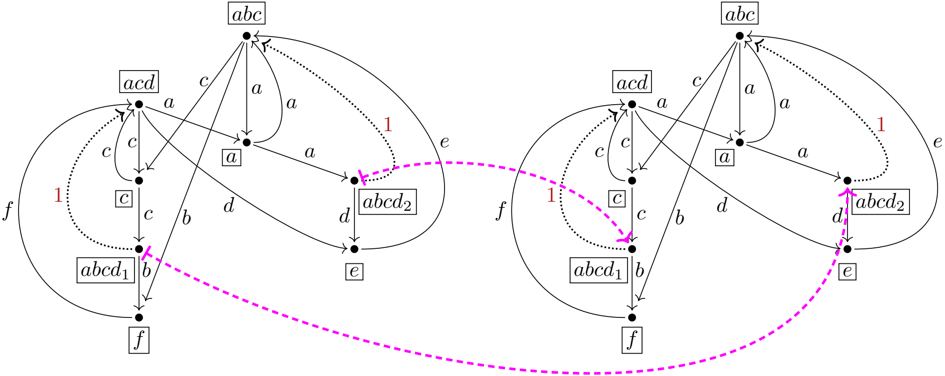


Local transfer function on 1-LTS

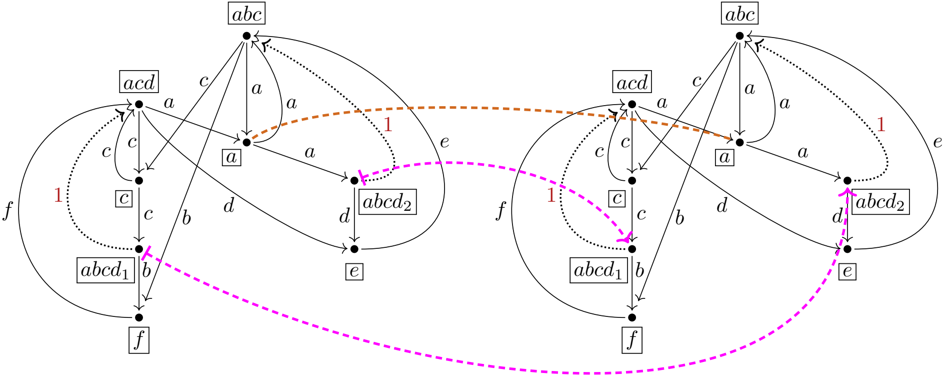




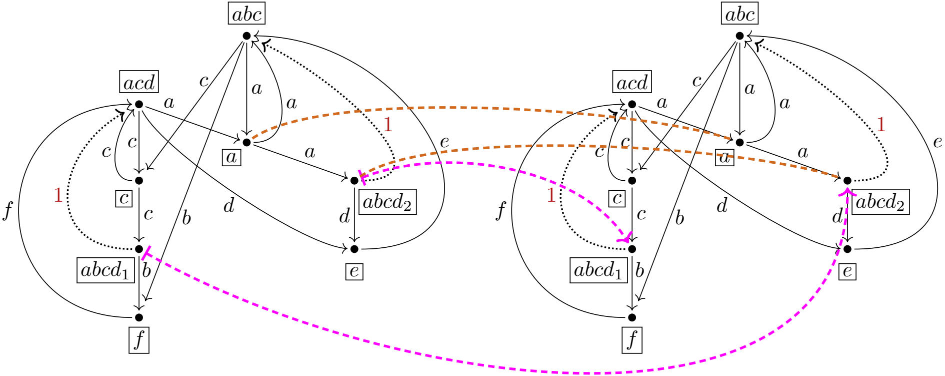
Local transfer function on 1-LTS



Local transfer function on 1-LTS



Local transfer function on 1-LTS



Summary

- fragments of bisimulations
  - bisimulating slices
    - sometimes useful to construct bisimulations
  - grounded bisimulation slices
    - a grounded bisimulation slice B on an LTS $\mathcal{L}$ gives rise to a bisimulation $\overline{\overline{B}} := B \cup =$ on $\mathcal{L}$.
- functional fragments of bisimulations
  - pullback of specifications via functional bisimulations
  - transfer functions $\hat{=}$ functional bisimulations
  - local-transfer functions $\hat{=}$ functional grounded bisimulation slices
    - a local-transfer function ϕ on an LTS $\mathcal{L}$ lifts to a transfer function $\widehat{\phi}$ on the elevation of $\mathrm{E}_{\mathrm{field}(\phi)}(\mathcal{L})$ of field(ϕ) above $\mathcal{L}$.
- application aimed at:
  - using local transfer functions on near-collapsed 1-LTSs
  - for transfer of specifications, to prove them equal in a proof system

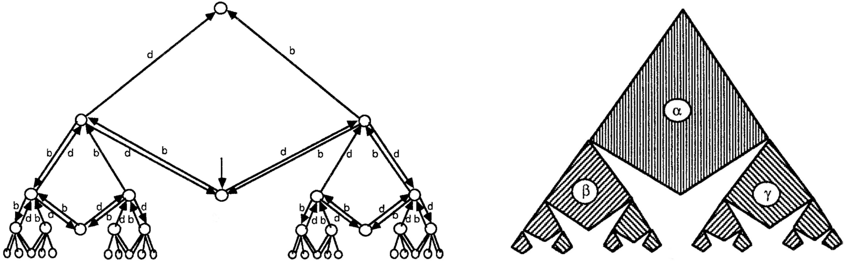
Summary

- fragments of bisimulations
  - bisimulating slices
    - sometimes useful to construct bisimulations
  - grounded bisimulation slices
    - a grounded bisimulation slice B on an LTS $\mathcal{L}$ gives rise to a bisimulation $\overline{\overline{B}} := B \cup =$ on $\mathcal{L}$.
- functional fragments of bisimulations
  - pullback of specifications via functional bisimulations
  - transfer functions $\hat{=}$ functional bisimulations
  - local-transfer functions $\hat{=}$ functional grounded bisimulation slices
    - a local-transfer function ϕ on an LTS $\mathcal{L}$ lifts to a transfer function $\widehat{\phi}$ on the elevation of $\mathrm{E}_{\mathrm{field}(\phi)}(\mathcal{L})$ of field(ϕ) above $\mathcal{L}$.
- application aimed at:
  - using local transfer functions on near-collapsed 1-LTSs
  - for transfer of specifications, to prove them equal in a proof system

Summary

- fragments of bisimulations
  - bisimulating slices
    - sometimes useful to construct bisimulations
  - grounded bisimulation slices
    - a grounded bisimulation slice B on an LTS $\mathcal{L}$ gives rise to a bisimulation $\overline{\overline{B}} := B \cup =$ on $\mathcal{L}$.
- functional fragments of bisimulations
  - pullback of specifications via functional bisimulations
  - transfer functions $\hat{=}$ functional bisimulations
  - local-transfer functions $\hat{=}$ functional grounded bisimulation slices
    - a local-transfer function ϕ on an LTS $\mathcal{L}$ lifts to a transfer function $\widehat{\phi}$ on the elevation of $\mathrm{E}_{\mathrm{field}(\phi)}(\mathcal{L})$ of field(ϕ) above $\mathcal{L}$.
- application aimed at:
  - using local transfer functions on near-collapsed 1-LTSs
  - for transfer of specifications, to prove them equal in a proof system

Bisimulating/bisimulation slices: origin and application



$$X = dY + bZ\ , \quad Y = b + bX + dYY\ , \quad Z = d + dX + bZZ$$

Example graph (concrete and abstract form) from:

Baeten, Bergstra, Klop: *Decidability of Bisimulation Equivalence for Processes Generating Context-Free Languages*, 1987

Bisimulating/bisimulation slices: origin and application

If for *each* pair α,β in the k-th slice such a copy α',β' exists, then the partial bisimulation R is called d-*sufficient*.

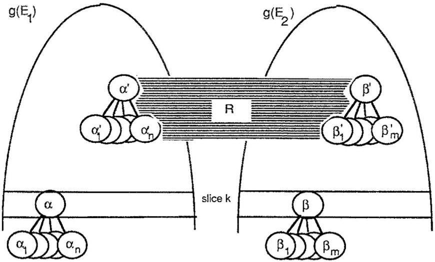


Figure 16

6.4. Let a partial bisimulation R as in 6.3 be given, which is sufficient. Then the *periodical continuation* of R is constructed as follows. Let α,β be as in 6.3. The partial bisimulation R is extended to $(\alpha_1 \cup ... \cup \alpha_n) \times (\beta_1 \cup ... \cup \beta_m)$ by copying the restriction of R to $(\alpha_1' \cup ... \cup \alpha_n') \times (\beta_1' \cup ... \cup \beta_m')$. This is done for all pairs α,β in slice k of $g(E_1)$, $g(E_2)$. It is

Baeten, Bergstra, Klop: *Decidability of Bisimulation Equivalence for Processes Generating Context-Free Languages*, 1987

Bisimulating/bisimulation slices: origin and application

If for *each* pair α,β in the k-th slice such a copy α',β' exists, then the partial bisimulation R is called d-*sufficient*.

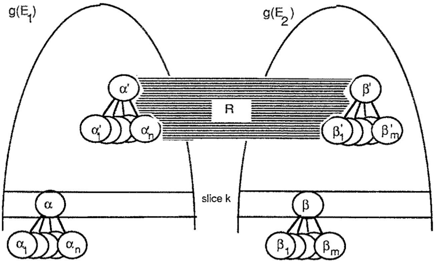


Figure 16

6.4. Let a partial bisimulation R as in 6.3 be given, which is sufficient. Then the *periodical continuation* of R is constructed as follows. Let α,β be as in 6.3. The partial bisimulation R is extended to $(\alpha_1 \cup ... \cup \alpha_n) \times (\beta_1 \cup ... \cup \beta_m)$ by copying the restriction of R to $(\alpha_1' \cup ... \cup \alpha_n') \times (\beta_1' \cup ... \cup \beta_m')$. This is done for all pairs α,β in slice k of $g(E_1)$, $g(E_2)$. It is

Baeten, Bergstra, Klop: *Decidability of Bisimulation Equivalence for Processes Generating Context-Free Languages*, 1987

Bisimulation, functional bisimulation

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$, for $i \in \{1,2\}$ be LTSs.

A *bisimulation between $\mathcal{L}_1$ and $\mathcal{L}_2$* is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ such that for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$\text{(forth)} \quad \forall v_1' \in S_1 \; \forall a \in A \; \big(v_1 \xrightarrow{a}_1 v_1' \implies \exists v_2' \in S_2 \big(v_2 \xrightarrow{a}_2 v_2' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \big) \,,$$

Bisimulation, functional bisimulation

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$, for $i \in \{1,2\}$ be LTSs.

A *bisimulation between $\mathcal{L}_1$ and $\mathcal{L}_2$* is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ such that for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$\text{(forth)} \quad \forall v_1' \in S_1 \, \forall a \in A \quad \left(v_1 \xrightarrow{a}_1 v_1' \implies \exists v_2' \in S_2 \left(v_2 \xrightarrow{a}_2 v_2' \land \langle v_1', v_2' \rangle \in \boldsymbol{B} \right) \right),$$

$$\text{(back)} \quad \forall v_2' \in S_2 \forall a \in A \quad \left(\exists v_1' \in S_1 \left(v_1 \xrightarrow{a}_1 v_1' \land \langle v_1', v_2' \rangle \in \boldsymbol{B} \right) \Longleftarrow v_2 \xrightarrow{a}_2 v_2' \right),$$

Bisimulation, functional bisimulation

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$, for $i \in \{1,2\}$ be LTSs.

A *bisimulation between $\mathcal{L}_1$ and $\mathcal{L}_2$* is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ such that for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$\text{(forth)} \quad \forall v_1' \in S_1 \, \forall a \in A \; \Big(v_1 \xrightarrow{a}_1 v_1' \implies \exists v_2' \in S_2 \big(v_2 \xrightarrow{a}_2 v_2' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \Big),$$

$$\text{(back)} \quad \forall v_2' \in S_2 \forall a \in A \; \Big(\exists v_1' \in S_1 \big(v_1 \xrightarrow{a}_1 v_1' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \impliedby v_2 \xrightarrow{a}_2 v_2' \Big),$$

$$\text{(termination)} \quad v_1 \downarrow_1 \iff v_2 \downarrow_2 .$$

Bisimulation, functional bisimulation

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$, for $i \in \{1,2\}$ be LTSs.

A *bisimulation between $\mathcal{L}_1$ and $\mathcal{L}_2$* is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ such that for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$\begin{array}{rl} \text{(forth)} & \forall v_1' \in S_1 \; \forall a \in A \\ & \quad \Big(v_1 \xrightarrow{a}_1 v_1' \implies \exists v_2' \in S_2 \big(v_2 \xrightarrow{a}_2 v_2' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \Big) \,, \\ \text{(back)} & \forall v_2' \in S_2 \forall a \in A \\ & \quad \Big(\exists v_1' \in S_1 \big(v_1 \xrightarrow{a}_1 v_1' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \impliedby v_2 \xrightarrow{a}_2 v_2' \Big) \,, \\ \text{(termination)} & v_1 {\downarrow_1} \iff v_2 {\downarrow_2} \,. \end{array}$$

We denote it by $\mathcal{L}_1 \leftharpoondown \boldsymbol{B} \rightharpoonup \mathcal{L}_2$.

Bisimulation, functional bisimulation

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$, for $i \in \{1,2\}$ be LTSs.

A *bisimulation between $\mathcal{L}_1$ and $\mathcal{L}_2$* is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ such that for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$\begin{array}{rl} \text{(forth)} & \forall v_1' \in S_1 \; \forall a \in A \\ & \big(v_1 \xrightarrow{a}_1 v_1' \implies \exists v_2' \in S_2 \big(v_2 \xrightarrow{a}_2 v_2' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \big) \,, \\ \text{(back)} & \forall v_2' \in S_2 \forall a \in A \\ & \big(\exists v_1' \in S_1 \big(v_1 \xrightarrow{a}_1 v_1' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \impliedby v_2 \xrightarrow{a}_2 v_2' \big) \,, \\ \text{(termination)} & v_1 \downarrow_1 \iff v_2 \downarrow_2 \,. \end{array}$$

We denote it by $\mathcal{L}_1 \leftharpoondown \boldsymbol{B} \rightharpoondown \mathcal{L}_2$.

A *functional bisimulation between $\mathcal{L}_1$ and $\mathcal{L}_2$* is a bisimulation that is the graph $\mathrm{gr}(\phi)$ of a partial function $\phi : S_1 \rightharpoonup S_2$. We write $\mathcal{L}_1 = \phi \rightharpoondown \mathcal{L}_2$.

Bisimulating/bisimulation slices

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$ for $i \in \{1, 2\}$ be LTSs.
A *bisimulating slice between $\mathcal{L}_1$, $\mathcal{L}_2$* is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ with active domain $W_1 := \mathrm{dom}_{\mathsf{act}}(\boldsymbol{B}) := \{x \mid \langle x, y \rangle \in \boldsymbol{B}\}$, active codomain $W_2 := \mathrm{cod}_{\mathsf{act}}(\boldsymbol{B}) := \{y \mid \langle x, y \rangle \in \boldsymbol{B}\}$ s.th. for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$\begin{aligned}
(\text{forth})_{\mathsf{s}} \quad & \forall a \in A\ \forall v_1' \in S_1 \\
& \quad \big(v_1 \xrightarrow{a}_1 v_1' \wedge \boxed{v_1' \in W_1} \\
& \qquad \Longrightarrow\ \exists v_2' \in S_2 \big(v_2 \xrightarrow{a}_2 v_2' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \big) ,
\end{aligned}$$

Bisimulating/bisimulation slices

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$ for $i \in \{1,2\}$ be LTSs.
A *bisimulating slice between* $\mathcal{L}_1$, $\mathcal{L}_2$ is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ with active domain $W_1 := \mathrm{dom}_{\mathsf{act}}(\boldsymbol{B}) := \{x \mid \langle x, y \rangle \in \boldsymbol{B}\}$, active codomain $W_2 := \mathrm{cod}_{\mathsf{act}}(\boldsymbol{B}) := \{y \mid \langle x, y \rangle \in \boldsymbol{B}\}$ s.th. for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$\begin{aligned}
(\text{forth})_{\mathsf{s}} \quad & \forall a \in A \; \forall v_1' \in S_1 \\
& \Big(v_1 \xrightarrow{a}_1 v_1' \wedge \boxed{v_1' \in W_1} \\
& \qquad \Longrightarrow \; \exists v_2' \in S_2 \big(v_2 \xrightarrow{a}_2 v_2' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \Big) \,, \\
(\text{back})_{\mathsf{s}} \quad & \forall a \in A \; \forall v_2' \in S_2 \\
& \Big(\exists v_1' \in S_1 \big(v_1 \xrightarrow{a}_1 v_1' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \\
& \qquad \Longleftarrow \; v_2 \xrightarrow{a}_2 v_2' \wedge \boxed{v_2' \in W_2} \Big) \,,
\end{aligned}$$

Bisimulating/bisimulation slices

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$ for $i \in \{1,2\}$ be LTSs.
A *bisimulating slice between $\mathcal{L}_1$, $\mathcal{L}_2$* is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ with active domain $W_1 := \mathrm{dom}_{\mathsf{act}}(\boldsymbol{B}) := \{x \mid \langle x, y \rangle \in \boldsymbol{B}\}$, active codomain $W_2 := \mathrm{cod}_{\mathsf{act}}(\boldsymbol{B}) := \{y \mid \langle x, y \rangle \in \boldsymbol{B}\}$ s.th. for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$\begin{aligned}
(\text{forth})_{\mathsf{s}} \quad & \forall a \in A \; \forall v_1' \in S_1 \\
& \Big(v_1 \xrightarrow{a}_1 v_1' \wedge \boxed{v_1' \in W_1} \\
& \qquad \Longrightarrow \; \exists v_2' \in S_2 \big(v_2 \xrightarrow{a}_2 v_2' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \Big) , \\
(\text{back})_{\mathsf{s}} \quad & \forall a \in A \; \forall v_2' \in S_2 \\
& \Big(\exists v_1' \in S_1 \big(v_1 \xrightarrow{a}_1 v_1' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \\
& \qquad \Longleftarrow \; v_2 \xrightarrow{a}_2 v_2' \wedge \boxed{v_2' \in W_2} \Big) , \\
(\text{termination})_{\mathsf{s}} \quad & v_1 \downarrow_1 \;\Longleftrightarrow\; v_2 \downarrow_2 .
\end{aligned}$$

Bisimulating/bisimulation slices

Definition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$ for $i \in \{1,2\}$ be LTSs.
A *bisimulating slice between* $\mathcal{L}_1$, $\mathcal{L}_2$ is a non-empty relation $\boldsymbol{B} \subseteq S_1 \times S_2$ with active domain $W_1 := \mathrm{dom}_{\mathsf{act}}(\boldsymbol{B}) := \{x \mid \langle x, y \rangle \in \boldsymbol{B}\}$, active codomain $W_2 := \mathrm{cod}_{\mathsf{act}}(\boldsymbol{B}) := \{y \mid \langle x, y \rangle \in \boldsymbol{B}\}$ s.th. for all $\langle v_1, v_2 \rangle \in \boldsymbol{B}$:

$$
\begin{aligned}
&(\text{forth})_{\mathsf{s}} \quad \forall a \in A\, \forall v_1' \in S_1 \\
&\qquad \Big(v_1 \xrightarrow{a}_1 v_1' \wedge \boxed{v_1' \in W_1} \\
&\qquad\qquad \Longrightarrow \exists v_2' \in S_2 \big(v_2 \xrightarrow{a}_2 v_2' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \big) \Big)\,,
\end{aligned}
$$

$$
\begin{aligned}
&(\text{back})_{\mathsf{s}} \quad \forall a \in A\, \forall v_2' \in S_2 \\
&\qquad \Big(\exists v_1' \in S_1 \big(v_1 \xrightarrow{a}_1 v_1' \wedge \langle v_1', v_2' \rangle \in \boldsymbol{B} \\
&\qquad\qquad \Longleftarrow v_2 \xrightarrow{a}_2 v_2' \wedge \boxed{v_2' \in W_2} \Big)\,,
\end{aligned}
$$

$$
(\text{termination})_{\mathsf{s}} \quad v_1{\downarrow_1} \iff v_2{\downarrow_2}\,.
$$

A *bisimulation slice between* $\mathcal{L}_1$ *and* $\mathcal{L}_2$ is a bisimulating slice between $\mathcal{L}_1$ and $\mathcal{L}_2$ that is contained in a bisimulation between $\mathcal{L}_1$ and $\mathcal{L}_2$.

Transfer functions: conditions

Proposition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$ for $i \in \{1, 2\}$ be LTSs.

A partial function $\phi : S_1 \rightharpoonup S_2$ with domain $W_1 := \mathrm{dom}(\phi)$ is a transfer function between $\mathcal{L}_1$ and $\mathcal{L}_2$ if and only if $W_1 \neq \varnothing$, and for all $s_1, s_1', s_2' \in S$ and $a \in A$:

$$s_1 \in W_1 \wedge s_1 \xrightarrow{a}_1 s_1' \implies \phi(s_1) \xrightarrow{a}_2 \phi(s_1') \ ,$$

Transfer functions: conditions

Proposition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$ for $i \in \{1,2\}$ be LTSs.

A partial function $\phi : S_1 \rightharpoonup S_2$ with domain $W_1 := \mathrm{dom}(\phi)$ is a transfer function between $\mathcal{L}_1$ and $\mathcal{L}_2$ if and only if $W_1 \neq \varnothing$, and for all $s_1, s_1', s_2' \in S$ and $a \in A$:

$$s_1 \in W_1 \wedge s_1 \xrightarrow{a}_1 s_1' \implies \phi(s_1) \xrightarrow{a}_2 \phi(s_1') \ ,$$

$$\exists s_1' \in S_1 \big(\, s_1 \xrightarrow{a}_1 s_1' \wedge \phi(s_1') = s_2' \, \big) \impliedby s_1 \in W_1 \wedge \phi(s_1) \xrightarrow{a}_2 s_2' \ ,$$

Transfer functions: conditions

Proposition

Let $\mathcal{L}_i = \langle S_i, A, \to_i, \downarrow_i \rangle$ for $i \in \{1,2\}$ be LTSs.

A partial function $\phi : S_1 \rightharpoonup S_2$ with domain $W_1 := \mathrm{dom}(\phi)$ is a transfer function between $\mathcal{L}_1$ and $\mathcal{L}_2$ if and only if $W_1 \neq \varnothing$, and for all $s_1, s_1', s_2' \in S$ and $a \in A$:

$$s_1 \in W_1 \wedge s_1 \xrightarrow{a}_1 s_1' \implies \phi(s_1) \xrightarrow{a}_2 \phi(s_1') \ ,$$

$$\exists s_1' \in S_1 \big(\, s_1 \xrightarrow{a}_1 s_1' \wedge \phi(s_1') = s_2' \, \big) \impliedby s_1 \in W_1 \wedge \phi(s_1) \xrightarrow{a}_2 s_2' \ ,$$

$$s_1 \in W_1 \implies \big(\, s_1{\downarrow_1} \iff \phi(s_1){\downarrow_2} \, \big) \ .$$

Grounded bisimulation slices

Definition

Let $\mathcal{L} = \langle S, A, \rightarrow, \downarrow \rangle$ be an LTS.

A *grounded* bisimulation slice *on* $\mathcal{L}$ is a bisimulating slice $\boldsymbol{B} \subseteq S \times S$ with $\mathrm{dom}_{\mathsf{act}}(\boldsymbol{B}) := W_1$, $\mathrm{cod}_{\mathsf{act}}(\boldsymbol{B}) := W_2$ such that for every $\langle v_1, v_2 \rangle \in \boldsymbol{B}$ holds:

(forth)$_{\mathsf{g}}$ $\quad \forall a \in A \, \forall v_1' \in S_1$

$$\left(v_1 \xrightarrow{a} v_1' \wedge \boxed{v_1' \notin W_1} \implies v_2 \xrightarrow{a} v_1' \wedge \boxed{v_1' \notin W_2} \right),$$

Grounded bisimulation slices

Definition

Let $\mathcal{L} = \langle S, A, \to, \downarrow \rangle$ be an LTS.

A *grounded* bisimulation slice *on* $\mathcal{L}$ is a bisimulating slice $\boldsymbol{B} \subseteq S \times S$ with $\mathrm{dom}_{\mathsf{act}}(\boldsymbol{B}) := W_1$, $\mathrm{cod}_{\mathsf{act}}(\boldsymbol{B}) := W_2$ such that for every $\langle v_1, v_2 \rangle \in \boldsymbol{B}$ holds:

$$(\text{forth})_{\mathsf{g}} \quad \forall a \in A \, \forall v_1' \in S_1 \; \Big(v_1 \xrightarrow{a} v_1' \wedge \boxed{v_1' \notin W_1} \implies v_2 \xrightarrow{a} v_1' \wedge \boxed{v_1' \notin W_2} \Big) ,$$

$$(\text{back})_{\mathsf{g}} \quad \forall a \in A \, \forall v_2' \in S_1 \; \Big(v_1 \xrightarrow{a} v_2' \wedge \boxed{v_2' \notin W_1} \impliedby v_2 \xrightarrow{a} v_2' \wedge \boxed{v_2' \notin W_2} \Big) .$$

Elevation of a vertex set above LTS

Definition

Let $\mathcal{L} = \langle S, A, \to, \downarrow\rangle$ be an LTS, and $W \subseteq S$ a subset of the vertices of $\mathcal{L}$.

The *elevation of W above $\mathcal{L}$* is the LTS $\mathsf{E}_W(\mathcal{L}) = \langle S_{\mathsf{E}_W}, A, \to_{\mathsf{E}_W}, \downarrow_{\mathsf{E}_W}\rangle$:

$$S_{\mathsf{E}_W} := S \times \{\mathbf{0}, \mathbf{1}\} \;,$$

$$\begin{aligned}\to_{\mathsf{E}_W} := \;& \big\{\langle\langle v_1, \mathbf{1}\rangle, a, \langle v_2, \mathbf{1}\rangle\rangle \;\big|\; \langle v_1, a, v_2\rangle \in \to \wedge\, a \in A \wedge v_2 \in W\big\} \\ \cup\; & \big\{\langle\langle v_1, \mathbf{1}\rangle, a, \langle v_2, \mathbf{0}\rangle\rangle \;\big|\; \langle v_1, a, v_2\rangle \in \to \wedge\, a \in A \wedge v_2 \notin W\big\} \\ \cup\; & \big\{\langle\langle v_1, \mathbf{0}\rangle, a, \langle v_2, \mathbf{0}\rangle\rangle \;\big|\; \langle v_1, a, v_2\rangle \in \to\big\} \;,\end{aligned}$$

$$\downarrow_{\mathsf{E}_W} := \big\{\langle v, i\rangle \;\big|\; v \in S,\, i \in \{\mathbf{0}, \mathbf{1}\}\,,\, v\downarrow\big\} \;.$$

Elevation of a vertex set above LTS

Definition

Let $\mathcal{L} = \langle S, A, \to, \downarrow \rangle$ be an LTS, and $W \subseteq S$ a subset of the vertices of $\mathcal{L}$.

The *elevation of W above $\mathcal{L}$* is the LTS $\mathsf{E}_W(\mathcal{L}) = \langle S_{\mathsf{E}_W}, A, \to_{\mathsf{E}_W}, \downarrow_{\mathsf{E}_W} \rangle$:

$$S_{\mathsf{E}_W} := S \times \{\mathbf{0}, \mathbf{1}\} \;,$$

$$\begin{aligned}\to_{\mathsf{E}_W} := \; & \big\{ \langle \langle v_1, \mathbf{1} \rangle, a, \langle v_2, \mathbf{1} \rangle \rangle \;\big|\; \langle v_1, a, v_2 \rangle \in \to \wedge\, a \in A \wedge v_2 \in W \big\} \\ \cup & \big\{ \langle \langle v_1, \mathbf{1} \rangle, a, \langle v_2, \mathbf{0} \rangle \rangle \;\big|\; \langle v_1, a, v_2 \rangle \in \to \wedge\, a \in A \wedge v_2 \notin W \big\} \\ \cup & \big\{ \langle \langle v_1, \mathbf{0} \rangle, a, \langle v_2, \mathbf{0} \rangle \rangle \;\big|\; \langle v_1, a, v_2 \rangle \in \to \big\} \;,\end{aligned}$$

$$\downarrow_{\mathsf{E}_W} := \big\{ \langle v, i \rangle \;\big|\; v \in S,\, i \in \{\mathbf{0}, \mathbf{1}\},\, v\downarrow \big\} \;.$$

Proposition (elevation above LTS functionally bisimilar with LTS)

Let $\mathcal{L} = \langle S, A, \to, \downarrow \rangle$ be an LTS, and $W \subseteq S$.

Then $\mathsf{E}_W(\mathcal{L}) =\!\!\pi_1\!\!\Rightarrow \mathcal{L}$ holds, where $\pi_1 : S \times \{\mathbf{0}, \mathbf{1}\} \to S,\ \langle v, \boldsymbol{i} \rangle \mapsto v$ projection function.