Branching Bisimulation Games

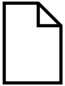
Jeroen Keiren
Open University of the Netherlands & Radboud Universiteit Nijmegen

Joint work with Tim Willemse (TU/e) and David de Frutos Escrig (UCM Madrid)

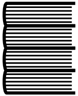
25 October 2016

Open Universiteit
www.ou.nl

Context



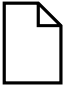
Specification



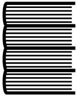
Implementation

Open Universiteit
www.ou.nl

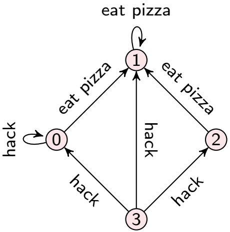
Context



Specification



Implementation



$L = \langle S, Act, \rightarrow \rangle$ Labelled Transition System

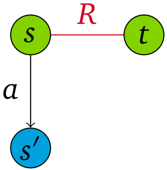
Strong Bisimulation

A strong bisimulation is a relation $R \subseteq S \times S$ on the states of an LTS $\langle S, Act, \rightarrow \rangle$ such that when $s\,R\,t$:

Open Universiteit
www.ou.nl

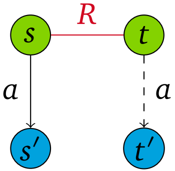
Strong Bisimulation

A strong bisimulation is a relation $R \subseteq S \times S$ on the states of an LTS $\langle S, Act, \rightarrow \rangle$ such that when $s\,R\,t$:



Strong Bisimulation

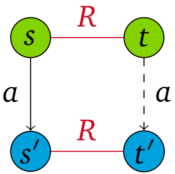
A strong bisimulation is a relation $R \subseteq S \times S$ on the states of an LTS $\langle S, Act, \rightarrow \rangle$ such that when $s\,R\,t$:



Open Universiteit
www.ou.nl

Strong Bisimulation

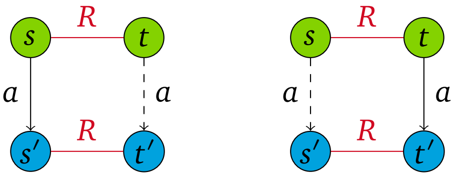
A strong bisimulation is a relation $R \subseteq S \times S$ on the states of an LTS $\langle S, Act, \rightarrow \rangle$ such that when $s\, R\, t$:



Open Universiteit
www.ou.nl

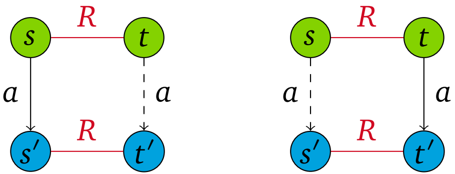
Strong Bisimulation

A strong bisimulation is a relation $R \subseteq S \times S$ on the states of an LTS $\langle S, Act, \rightarrow \rangle$ such that when $s\,R\,t$:



Strong Bisimulation

A strong bisimulation is a relation $R \subseteq S \times S$ on the states of an LTS $\langle S, Act, \rightarrow \rangle$ such that when $s\ R\ t$:



States s, t are bisimilar ($s \leftrightarrow t$) iff $s\ R\ t$ for some bisimulation R

Open Universiteit
www.ou.nl

Strong Bisimulation Games

Stirling's bisimulation game

- Ehrenfeucht-Fraïssé game
- player Spoiler (S) tries to disprove bisimilarity of s and t
- player Duplicator (D) tries to prove bisimilarity of s and t
- S wins all plays in which D 'gets stuck'
- D wins all other plays, i.e. both infinite plays and all plays in which S 'gets stuck'
- game is played in rounds (*ad infinitum* if possible)

Strong Bisimulation Games

Round starting in $[(s,t)]$:

1. S moves from configuration $[(s,t)]$ by:
   - selecting $s \xrightarrow{a} s'$ and moving to $\langle (s,t),(a,s') \rangle$, *or*
   - selecting $t \xrightarrow{a} t'$ and moving to $\langle (t,s),(a,t') \rangle$
2. D responds from configuration $\langle (u,v),(a,u') \rangle$ by:
   - moving $v \xrightarrow{a} v'$ and continue in configuration $[(u',v')]$.

$s \Leftrightarrow t$ iff Duplicator has a strategy to win all plays starting in (s,t)

Open Universiteit
www.ou.nl

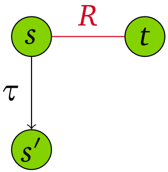
Branching Bisimulation

A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s \, R \, t$:

Open Universiteit
www.ou.nl

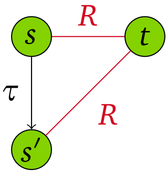
Branching Bisimulation

A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s\,R\,t$:



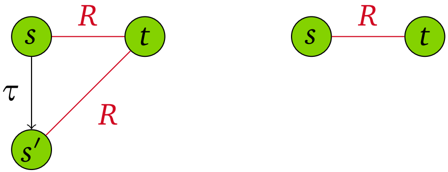
Branching Bisimulation

A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s\,R\,t$:



Branching Bisimulation

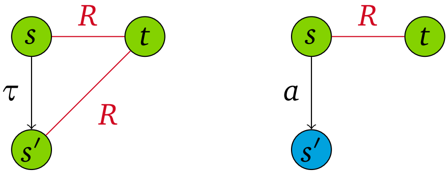
A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s\,R\,t$:



Open Universiteit
www.ou.nl

Branching Bisimulation

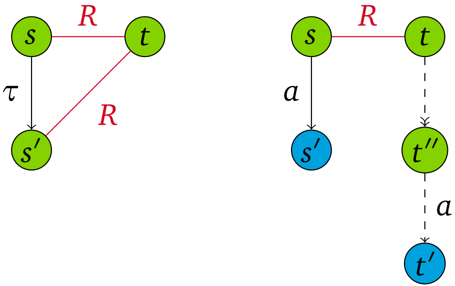
A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s \, R \, t$:



Open Universiteit
www.ou.nl

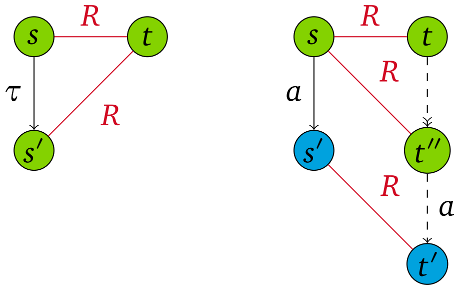
Branching Bisimulation

A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s \, R \, t$:



Branching Bisimulation

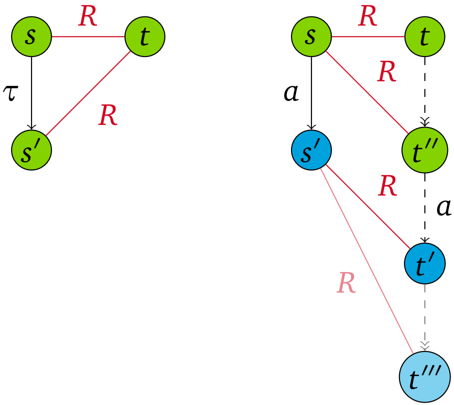
A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s\,R\,t$:



Open Universiteit
www.ou.nl

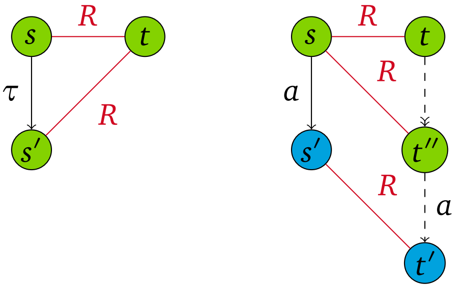
Branching Bisimulation

A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s\,R\,t$:



Branching Bisimulation

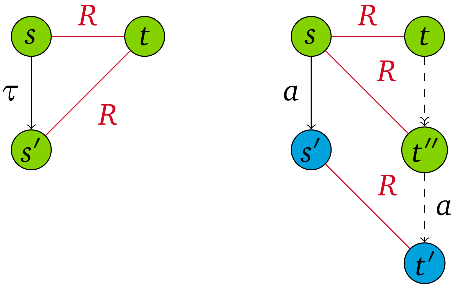
A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s \, R \, t$:



+ symmetric cases

Branching Bisimulation

A branching bisimulation is a relation $R \subseteq S \times S$ so that when $s\,R\,t$:

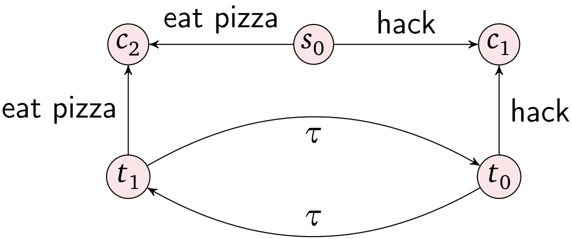


+ symmetric cases

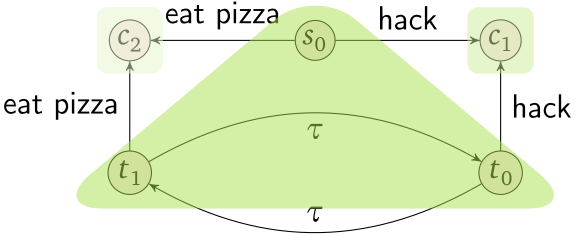
s, t are branching bisimilar ($s \leftrightarrows_b t$) iff $s\,R\,t$ for some bb. R

Open Universiteit
www.ou.nl

Branching Bisimulation Example



Branching Bisimulation Example



Branching Bisimulation Games

An attempt by Bulychev et al.

- S moves from configuration $[(s,t)]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to $\langle (s,t),(a,s')\rangle$, *or*
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle (t,s),(a,t')\rangle$
- D responds from a configuration $\langle (u,v),(a,u')\rangle$ by:
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v')]$, or

S wins all plays in which D gets stuck, D wins all other plays

Open Universiteit
www.ou.nl

Branching Bisimulation Games

An attempt by Bulychev et al.

- S moves from configuration $[(s,t)]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to $\langle(s,t),(a,s')\rangle$, *or*
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle(t,s),(a,t')\rangle$
- D responds from a configuration $\langle(u,v),(a,u')\rangle$ by:
  - not moving if $a = \tau$ and propose configuration $[(u',v)]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v')]$, or

S wins all plays in which D gets stuck, D wins all other plays

Open Universiteit
www.ou.nl

Branching Bisimulation Games

An attempt by Bulychev et al.

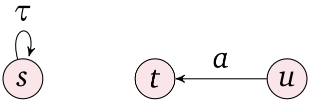
- S moves from configuration $[(s,t)]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to $\langle(s,t),(a,s')\rangle$, *or*
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle(t,s),(a,t')\rangle$
- D responds from a configuration $\langle(u,v),(a,u')\rangle$ by:
  - not moving if $a = \tau$ and propose configuration $[(u',v)]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v')]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v')]$

S wins all plays in which D gets stuck, D wins all other plays

Open Universiteit
www.ou.nl

Branching Bisimulation Games

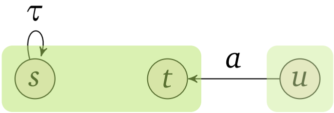
Problem with Bulychev's Definition



$$u \overset{?}{\leftrightarroweq}_b s$$

Branching Bisimulation Games

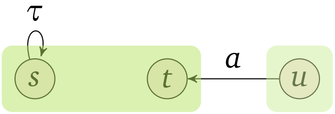
Problem with Bulychev's Definition



$$u \overset{?}{\leftrightarrow}_b s$$

Branching Bisimulation Games

Problem with Bulychev's Definition

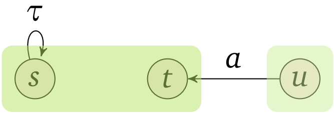


$u \overset{?}{\leftrightarrow}_b s$

- $[(u,s)]$
  - $\rightarrow \langle (u,s),(a,t)\rangle \rightarrow [(u,s)] \rightarrow \cdots$
  - $\rightarrow \langle (s,u),(\tau,s)\rangle \rightarrow [(s,u)] \rightarrow \cdots$

Branching Bisimulation Games

Problem with Bulychev's Definition



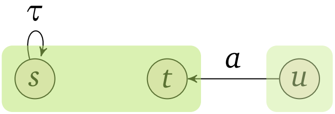
$u \overset{?}{\leftrightarrow}_b s$

- $[(u, s)]$
  - $\rightarrow \langle (u, s), (a, t) \rangle \rightarrow [(u, s)] \rightarrow \cdots$
  - $\rightarrow \langle (s, u), (\tau, s) \rangle \rightarrow [(s, u)] \rightarrow \cdots$

D wins, even though $u \not\leftrightarrow_b s$!

Branching Bisimulation Games

Problem with Bulychev's Definition



$u \overset{?}{\underline{\leftrightarrow}}_b s$

- $[(u,s)]$
  - $\rightarrow \langle (u,s),(a,t) \rangle \rightarrow [(u,s)] \rightarrow \cdots$
  - $\rightarrow \langle (s,u),(\tau,s) \rangle \rightarrow [(s,u)] \rightarrow \cdots$

D wins, even though $u \not\underline{\leftrightarrow}_b s$!

Definition only works for LTS without divergence

Branching Bisimulation Games

- observation: after one τ-step by Duplicator, we forget what Duplicator was mimicking

Open Universiteit
www.ou.nl

Branching Bisimulation Games

- observation: after one τ-step by Duplicator, we forget what Duplicator was mimicking
- intuition: Duplicator wishes to show she can meet every challenge

Open Universiteit
www.ou.nl

Branching Bisimulation Games

- observation: after one τ-step by Duplicator, we forget what Duplicator was mimicking
- intuition: Duplicator wishes to show she can meet every challenge
- idea: keep track of challenge that still needs to be fulfilled

Open Universiteit
www.ou.nl

Branching Bisimulation Games

- observation: after one τ-step by Duplicator, we forget what Duplicator was mimicking
- intuition: Duplicator wishes to show she can meet every challenge
- idea: keep track of challenge that still needs to be fulfilled
- reward Duplicator when she meets challenge, and

Open Universiteit
www.ou.nl

Branching Bisimulation Games

- observation: after one τ-step by Duplicator, we forget what Duplicator was mimicking
- intuition: Duplicator wishes to show she can meet every challenge
- idea: keep track of challenge that still needs to be fulfilled
- reward Duplicator when she meets challenge, and
- punish Spoiler when he changes challenge, by rewarding Duplicator

Branching Bisimulation Games

- observation: after one τ-step by Duplicator, we forget what Duplicator was mimicking
- intuition: Duplicator wishes to show she can meet every challenge
- idea: keep track of challenge that still needs to be fulfilled
- reward Duplicator when she meets challenge, and
- punish Spoiler when he changes challenge, by rewarding Duplicator

We play on configurations with

- challenges $c \in (A \times S) \cup \{\dagger\}$
- rewards $r \in \{*, \checkmark\}$

Open Universiteit
www.ou.nl

Branching bisimulation games

- S moves from configuration $[(s,t) \quad]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to
    - $\langle (s,t),(a,s') \quad \rangle$
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle (t,s),(a,t') \quad \rangle$
- D responds from a configuration $\langle (u,v),(a,u') \quad \rangle$ by:
  - not moving if $a = \tau$ and propose configuration $[(u',v) \quad]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v') \quad]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v') \quad]$

D wins a play if S gets stuck, and all infinite plays

Branching bisimulation games

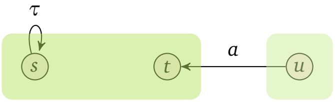
- S moves from configuration $[(s,t),c,r]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to
    - $\langle(s,t),(a,s'),*\rangle$
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle(t,s),(a,t'),*\rangle$
- D responds from a configuration $\langle(u,v),(a,u'),r\rangle$ by:
  - not moving if $a=\tau$ and propose configuration $[(u',v),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v'),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v'),(a,u'),*]$

D wins a play if S gets stuck, or she gets infinitely many $\checkmark$ rewards

Open Universiteit
www.ou.nl

Branching Bisimulation Games

Rewards for Duplicator

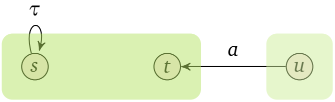


$u \overset{?}{\leftrightarrow}_b s$

Open Universiteit
www.ou.nl

Branching Bisimulation Games

Rewards for Duplicator



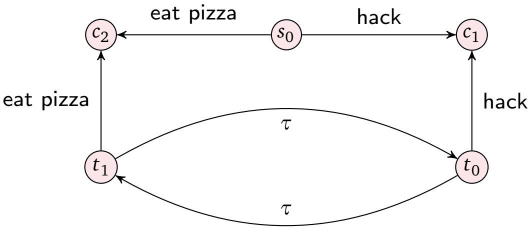
$u \overset{?}{\leftrightarrow}_b s$

- $[(u,s),\dagger,*] \rightarrow \langle(u,s),(a,t),*\rangle \rightarrow [(u,s),(a,t),*] \rightarrow \cdots$

Open Universiteit
www.ou.nl

Branching Bisimulation Games

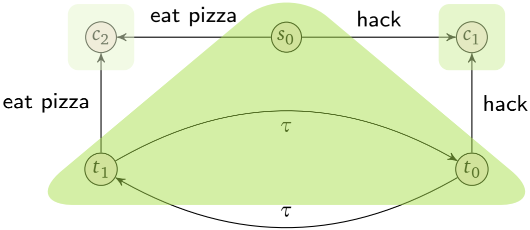
Not quite there yet...



Open Universiteit
www.ou.nl

Branching Bisimulation Games

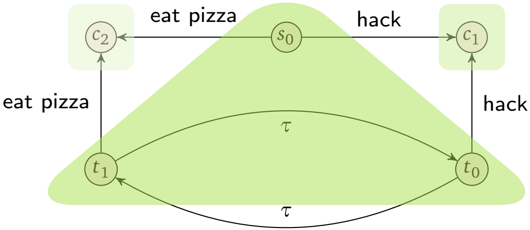
Not quite there yet...



Open Universiteit
www.ou.nl

Branching Bisimulation Games

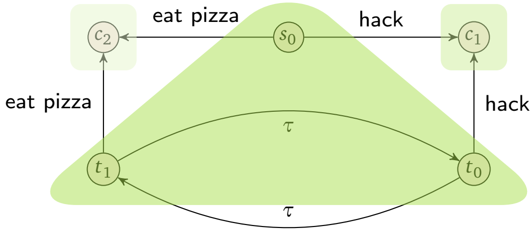
Not quite there yet...



Consider $[(s_0, t_1), \dagger, *]$, and assume S alternates challenge between eat pizza and hack:

Branching Bisimulation Games

Not quite there yet...



Consider $[(s_0, t_1), \dagger, *]$, and assume S alternates challenge between eat pizza and hack:

$[(s_0, t_1), \dagger, *] \to \langle (s_0, t_1), (\text{hack}, c_1), * \rangle \to [(s_0, t_0), (\text{hack}, c_1), *] \to \langle (s_0, t_0), (\text{eat pizza}, c_1), * \rangle \to [(s_0, t_1), (\text{eat pizza}, c_1), *] \to \cdots$

Branching bisimulation games

Punishing Spoiler

- S moves from configuration $[(s,t),c,r]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to
    - $\langle (s,t),(a,s'),*\rangle$
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle (t,s),(a,t'),*\rangle$
- D responds from a configuration $\langle (u,v),(a,u'),r\rangle$ by:
  - not moving if $a=\tau$ and propose configuration $[(u',v),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v'),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v'),(a,u'),*]$

D wins a play if S gets stuck, or she gets infinitely many $\checkmark$ rewards

Open Universiteit
www.ou.nl

Branching bisimulation games

Punishing Spoiler

- S moves from configuration $[(s,t),c,r]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to
    - $\langle (s,t),(a,s'),* \rangle$
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle (t,s),(a,t'),* \rangle$
- D responds from a configuration $\langle (u,v),(a,u'),r \rangle$ by:
  - not moving if $a = \tau$ and propose configuration $[(u',v),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v'),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v'),(a,u'),*]$

D wins a play if S gets stuck, or she gets infinitely many $\checkmark$ rewards

Open Universiteit
www.ou.nl

Branching bisimulation games

Punishing Spoiler

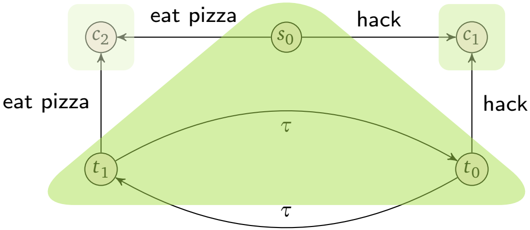
- S moves from configuration $[(s,t),c,r]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to
    - $\langle(s,t),(a,s'),*\rangle$ if $c=(a,s')$ or $c=\dagger$, *and to*
    - $\langle(s,t),(a,s'),\checkmark\rangle$, otherwise; *or*
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle(t,s),(a,t'),\checkmark\rangle$
- D responds from a configuration $\langle(u,v),(a,u'),r\rangle$ by:
  - not moving if $a=\tau$ and propose configuration $[(u',v),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v'),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v'),(a,u'),*]$

D wins a play if S gets stuck, or she gets infinitely many $\checkmark$ rewards

Open Universiteit
www.ou.nl

Branching Bisimulation Games

Example

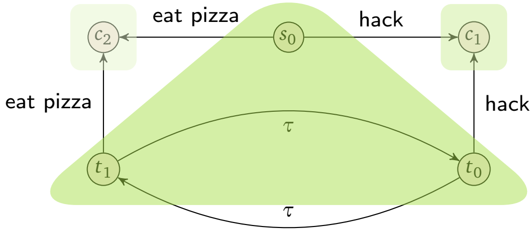


Reconsider the case where S alternates challenge between eat pizza and hack:

Open Universiteit
www.ou.nl

Branching Bisimulation Games

Example



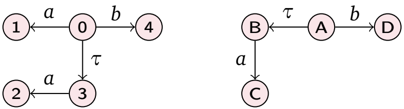
Reconsider the case where S alternates challenge between eat pizza and hack:

$[(s_0, t_1), \dagger, *] \to \langle (s_0, t_1), (\text{hack}, c_1), * \rangle \to [(s_0, t_0), (\text{hack}, c_1), *] \to$
$\langle (s_0, t_0), (\text{eat pizza}, c_1), \checkmark \rangle \to [(s_0, t_1), (\text{eat pizza}, c_1), *] \to \cdots$

Open Universiteit
www.ou.nl

Branching Bisimulation Games

Example

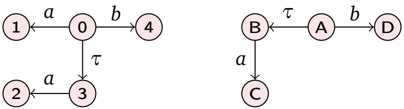


Spoiler (computer) plays against Duplicator (you) showing $0 \not\leftrightarrow_b A$:

Open Universiteit
www.ou.nl

Branching Bisimulation Games

Example

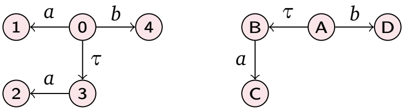


Spoiler (computer) plays against Duplicator (you) showing $0 \not\leftrightarrow_b A$:

`Spoiler challenges` $0 \xrightarrow{a} 1$

Branching Bisimulation Games

Example



Spoiler (computer) plays against Duplicator (you) showing $0 \not\Leftrightarrow_b A$:

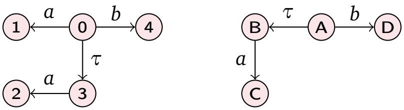
`Spoiler challenges` $0 \xrightarrow{a} 1$

`Your response:` $A \xrightarrow{\tau} B$`; you continue playing from` $((0, B), (a, 1))$

Open Universiteit
www.ou.nl

Branching Bisimulation Games

Example

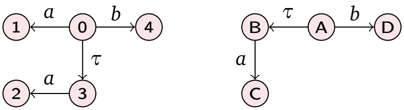


Spoiler (computer) plays against Duplicator (you) showing $0 \not\leftrightarrow_b A$:

`Spoiler challenges` $0 \xrightarrow{a} 1$
`Your response:` $A \xrightarrow{\tau} B$`; you continue playing from` $((0,B),(a,1))$
`Spoiler drops challenge` $(a,1)$ `and challenges` $0 \xrightarrow{b} 4$`. You earn` ✓

Branching Bisimulation Games

Example



Spoiler (computer) plays against Duplicator (you) showing $0 \not\leftrightarrow_b A$:

Spoiler challenges $0 \xrightarrow{a} 1$

Your response: $A \xrightarrow{\tau} B$; you continue playing from $((0,B),(a,1))$

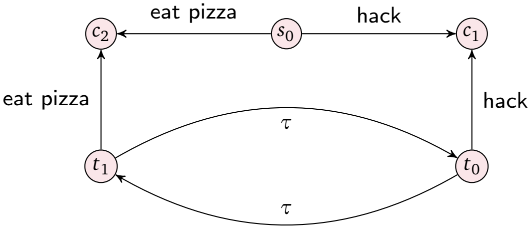
Spoiler drops challenge $(a,1)$ and challenges $0 \xrightarrow{b} 4$. You earn ✓

You cannot respond. You lose.

Open Universiteit
www.ou.nl

Extensions

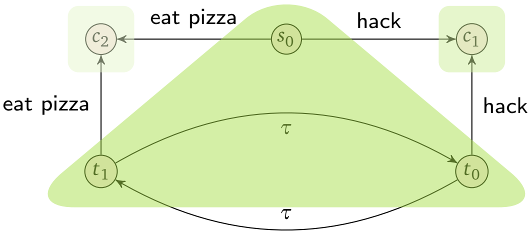
Branching Bisimulation with Explicit Divergence



Open Universiteit
www.ou.nl

Extensions

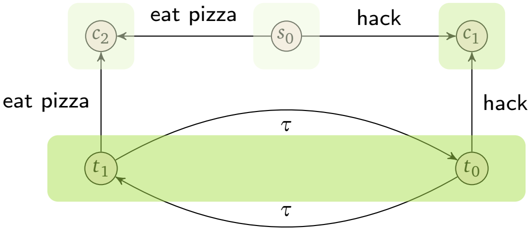
Branching Bisimulation with Explicit Divergence



Open Universiteit
www.ou.nl

Extensions

Branching Bisimulation with Explicit Divergence



Open Universiteit
www.ou.nl

Extensions

Branching Bisimulation with Explicit Divergence

- S moves from configuration $[(s,t),c,r]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to
    - $\langle (s,t),(a,s'),* \rangle$ if $c = (a,s')$ or $c = \dagger$, *and to*
    - $\langle (s,t),(a,s'),\checkmark \rangle$, otherwise; *or*
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle (t,s),(a,t'),\checkmark \rangle$
- D responds from a configuration $\langle (u,v),(a,u'),r \rangle$ by:
  - not moving if $a = \tau$ and propose configuration $[(u',v),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v'),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v'),(a,u'),*]$

D wins a play if S gets stuck, or she gets infinitely many $\checkmark$ rewards

Open Universiteit
www.ou.nl

Extensions

Branching Bisimulation with Explicit Divergence

- S moves from configuration $[(s,t),c,r]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to
    - $\langle(s,t),(a,s'),*\rangle$ if $c=(a,s')$ or $c=\dagger$, *and to*
    - $\langle(s,t),(a,s'),\checkmark\rangle$, otherwise; *or*
  - selecting $t \xrightarrow{a} t'$ and moving to $\langle(t,s),(a,t'),\checkmark\rangle$
- D responds from a configuration $\langle(u,v),(a,u'),r\rangle$ by:
  - not moving if $a=\tau$ and propose configuration $[(u',v),\dagger,*]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v'),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v'),(a,u'),*]$

D wins a play if S gets stuck, or she gets infinitely many $\checkmark$ rewards

Open Universiteit
www.ou.nl

Extensions

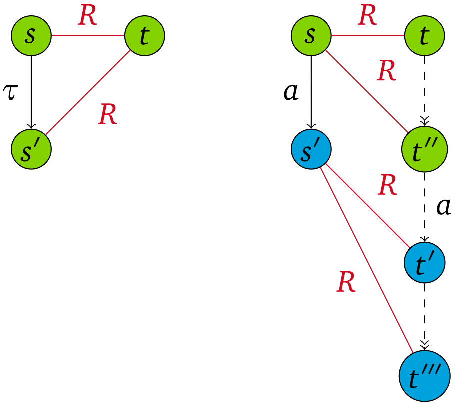
Branching Simulation

- S moves from configuration $[(s,t),c,r]$ by:
  - selecting $s \xrightarrow{a} s'$ and moving to
    - $\langle (s,t),(a,s'),* \rangle$ if $c = (a,s')$ or $c = \dagger$, *and to*
    - $\langle (s,t),(a,s'),\checkmark \rangle$, otherwise; *or*
  - ~~selecting $t \xrightarrow{a} t'$ and moving to $\langle (t,s),(a,t'),\checkmark \rangle$~~
- D responds from a configuration $\langle (u,v),(a,u'),r \rangle$ by:
  - not moving if $a = \tau$ and propose configuration $[(u',v),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{a} v'$ if available and continue in $[(u',v'),\dagger,\checkmark]$, or
  - moving $v \xrightarrow{\tau} v'$ if possible and continue in $[(u,v'),(a,u'),*]$

D wins a play if S gets stuck, or she gets infinitely many $\checkmark$ rewards

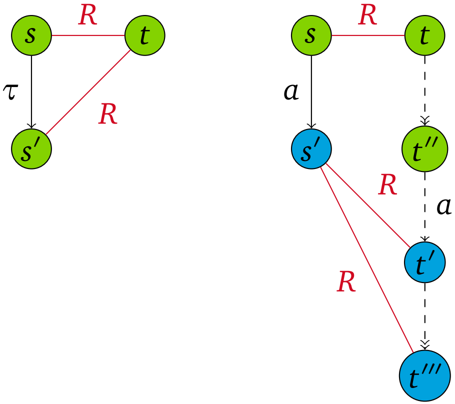
Extensions

Other weak behavioural equivalences



Extensions

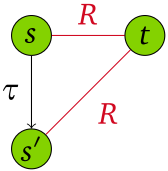
Other weak behavioural equivalences

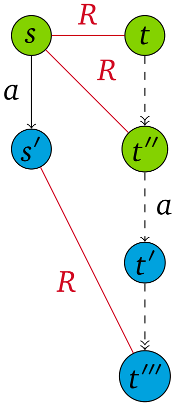


delay bisimulation

Extensions

Other weak behavioural equivalences



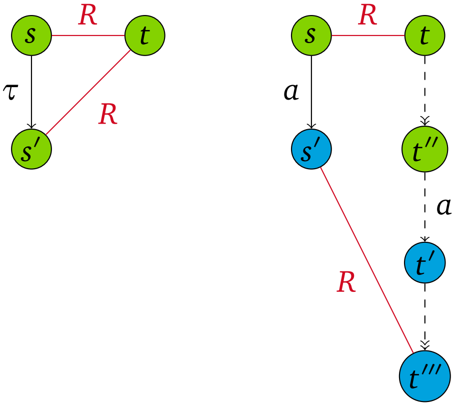


delay bisimulation η-bisimulation

Open Universiteit
www.ou.nl

Extensions

Other weak behavioural equivalences

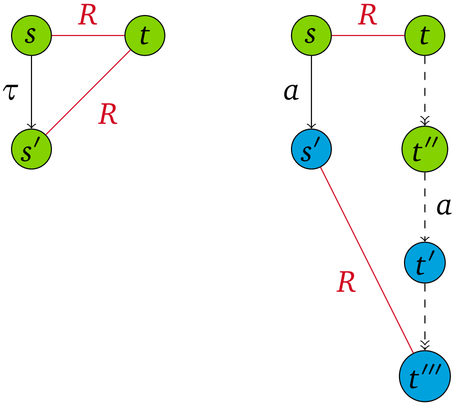


delay bisimulation η-bisimulation   weak bisimulation

Open Universiteit
www.ou.nl

Extensions

Other weak behavioural equivalences



delay bisimulation η-bisimulation   weak bisimulation

Summary

- Presented games for:
  - Branching bisimulation
  - Divergence preserving branching bisimulation
  - Branching simulation
- Require no preprocessing of input LTS
- Spoiler's winning strategy explains why games are not (bi)similar; this enables debugger-like applications

Open Universiteit
www.ou.nl

Future work

- Interactive implementation of our games (proof-of-concept available)

Open Universiteit
www.ou.nl

Future work

- Interactive implementation of our games (proof-of-concept available)
- Applications in education?

Thank you