Structure Preserving Bisimilarity,

Supporting an Operational Petri Net Semantics of CCSP

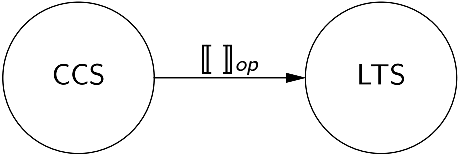
Rob van Glabbeek

NICTA, Sydney, Australia

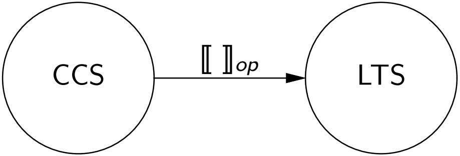
University of New South Wales, Sydney, Australia

September 2015

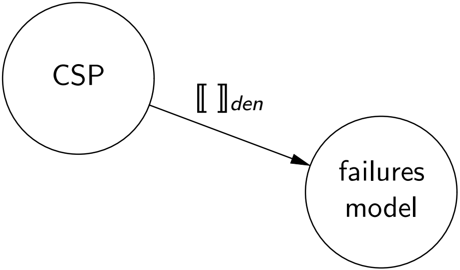
Milner:



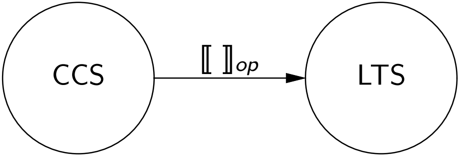
Milner:



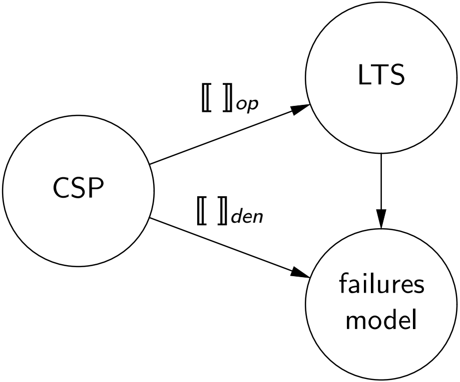
Hoare:



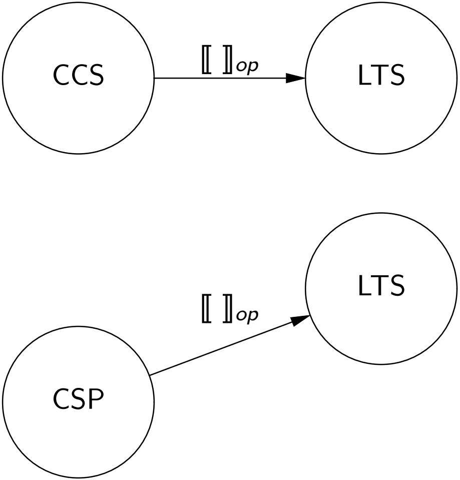
Milner:



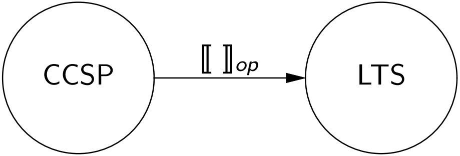
Hoare:



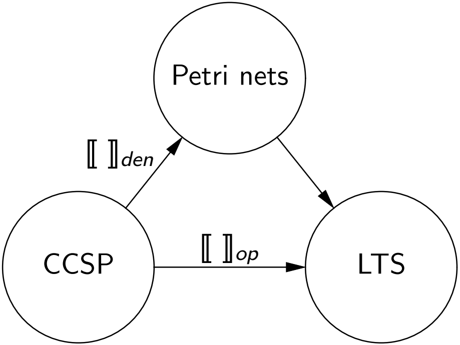
[Olderog & Hoare '86]


CCS
⟦ ⟧op
LTS
LTS
⟦ ⟧op
CSP

Nielsen:
Olderog:

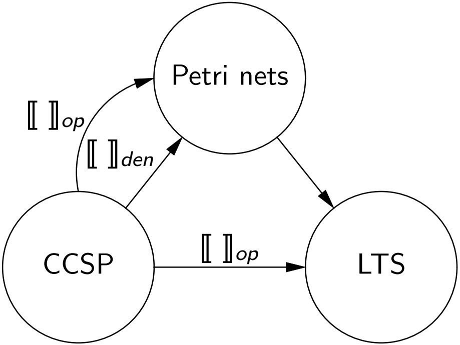


Goltz & Mycroft
Winskel
van Glabbeek & Vaandrager:



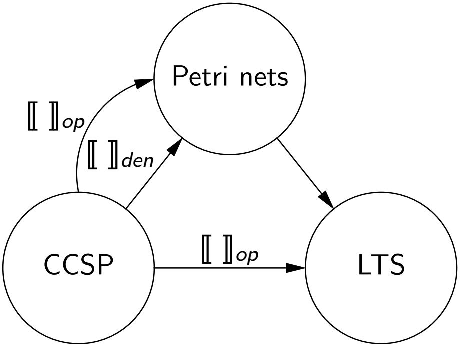
but no treatment of recursion.

Degano, De Nicola and Montanari:



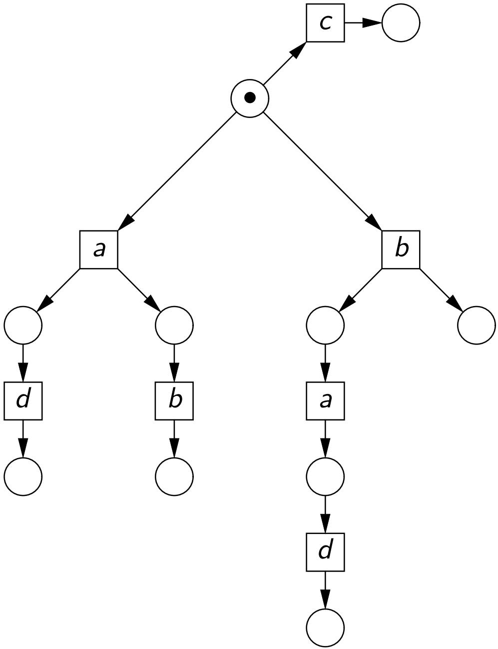
including a treatment of recursion.

Degano, De Nicola and Montanari:

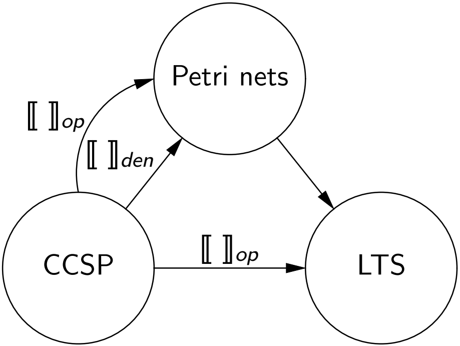


including a treatment of recursion.
But initial concurrency is not respected.

$$[\![(ad \| b) + c]\!]^{\mathrm{DDM}}_{op} =$$

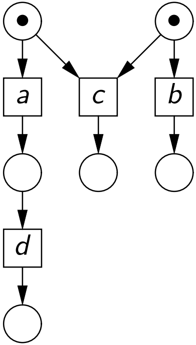


Olderog, 1987:

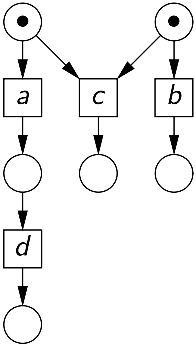


including a treatment of recursion.
Concurrency is fully respected.

$$[\![(ad \| b) + c]\!]_{op}^{\mathrm{Old}} =$$



$$[\![(ad \| b) + c]\!]^{\mathrm{Old}}_{op} =$$



This is the same result as for the denotational Petri net semantics found in the literature.

How to formalise the statement that Olderog's operational semantics respects concurrency?

How to formalise the statement that Olderog's operational semantics respects concurrency?

We use the well-known fact that the denotational semantics from the literature respects concurrency, at least for those expressions P for which it is defined (the recursion-free ones).

How to formalise the statement that Olderog's operational semantics respects concurrency?

We use the well-known fact that the denotational semantics from the literature respects concurrency, at least for those expressions P for which it is defined (the recursion-free ones).

So we want to prove: $[\![P]\!]_{op} = [\![P]\!]_{den}$.

How to formalise the statement that Olderog's operational semantics respects concurrency?
We use the well-known fact that the denotational semantics from the literature respects concurrency, at least for those expressions P for which it is defined (the recursion-free ones).
So we want to prove: $[\![P]\!]_{op} = [\![P]\!]_{den}$.
But this fails in the same way as the operational versus denotational semantics of CCSP in terms of labelled transition systems:

How to formalise the statement that Olderog's operational semantics respects concurrency?
We use the well-known fact that the denotational semantics from the literature respects concurrency, at least for those expressions P for which it is defined (the recursion-free ones).
So we want to prove: $[\![P]\!]_{op} = [\![P]\!]_{den}$.
But this fails in the same way as the operational versus denotational semantics of CCSP in terms of labelled transition systems: $[\![a + a]\!]_{op} \neq [\![a + a]\!]_{den}$.

How to formalise the statement that Olderog's operational semantics respects concurrency?

We use the well-known fact that the denotational semantics from the literature respects concurrency, at least for those expressions P for which it is defined (the recursion-free ones).

So we want to prove: $[\![P]\!]_{op} = [\![P]\!]_{den}$.

But this fails in the same way as the operational versus denotational semantics of CCSP in terms of labelled transition systems: $[\![a + a]\!]_{op} \neq [\![a + a]\!]_{den}$.

So instead one shows $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for a suitable relation $\approx$.

Showing agreement between an operational and denotational net semantics of CCSP

Aim: propose a relation $\approx$ between nets, and show $[\![P]\!]_{op} \approx [\![P]\!]_{den}$.

Showing agreement between an operational and denotational net semantics of CCSP

Aim: propose a relation $\approx$ between nets, and show $[\![P]\!]_{op} \approx [\![P]\!]_{den}$.

Which requirements to impose on $\approx$?

Showing agreement between an operational and denotational net semantics of CCSP

Aim: propose a relation $\approx$ between nets, and show $[\![P]\!]_{op} \approx [\![P]\!]_{den}$.

Which requirements to impose on $\approx$?

1. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Showing agreement between an operational and denotational net semantics of CCSP

Aim: propose a relation $\approx$ between nets, and show $[\![P]\!]_{op} \approx [\![P]\!]_{den}$.

Which requirements to impose on $\approx$?

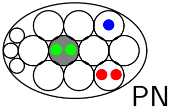
1. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.
2. $\approx$ respects concurrency.

Showing agreement between an operational and denotational net semantics of CCSP

Aim: propose a relation $\approx$ between nets, and show $[\![P]\!]_{op} \approx [\![P]\!]_{den}$.

Which requirements to impose on $\approx$?

1. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.
2. $\approx$ respects concurrency.
3. $\approx$ is a congruence relation for CCSP.



$R = P \| Q$

Showing agreement between an operational and denotational net semantics of CCSP

Olderog proposes a relation $\equiv$ between nets, called *strong bisimulation*.

Showing agreement between an operational and denotational net semantics of CCSP

Olderog proposes a relation $\equiv$ between nets, called *strong bisimulation*.

1. $[\![P]\!]_{op} \equiv [\![P]\!]_{den}$ for any P for which both sides are defined.

Showing agreement between an operational and denotational net semantics of CCSP

Olderog proposes a relation $\equiv$ between nets, called *strong bisimulation*.

1. $[\![P]\!]_{op} \equiv [\![P]\!]_{den}$ for any P for which both sides are defined.
2. $\equiv$ respects concurrency.

Showing agreement between an operational and denotational net semantics of CCSP

Olderog proposes a relation $\equiv$ between nets, called *strong bisimulation*.

1. $[\![P]\!]_{op} \equiv [\![P]\!]_{den}$ for any P for which both sides are defined.
2. $\equiv$ respects concurrency.
3. $\equiv$ has the typical congruence properties.

Showing agreement between an operational and denotational net semantics of CCSP

Olderog proposes a relation $\equiv$ between nets, called *strong bisimulation*.

1. $[\![P]\!]_{op} \equiv [\![P]\!]_{den}$ for any P for which both sides are defined.
2. $\equiv$ respects concurrency.
3. $\equiv$ has the typical congruence properties.

But it fails to be transitive!

Showing agreement between an operational and denotational net semantics of CCSP

Olderog proposes a relation $\equiv$ between nets, called *strong bisimulation*.

1. $[\![P]\!]_{op} \equiv [\![P]\!]_{den}$ for any P for which both sides are defined.
2. $\equiv$ respects concurrency.
3. $\equiv$ has the typical congruence properties.

But it fails to be transitive!
This is not a problem, because $\equiv$ can be seen as just a tool to proof things about *causal equivalence* $\equiv_{caus}$.
$N_1 \equiv_{caus} N_2$ if they have the same *processes* or *causal nets*.

Showing agreement between an operational and denotational net semantics of CCSP

Olderog proposes a relation $\equiv$ between nets, called *strong bisimulation*.

1. $[\![P]\!]_{op} \equiv [\![P]\!]_{den}$ for any P for which both sides are defined.
2. $\equiv$ respects concurrency.
3. $\equiv$ has the typical congruence properties.

But it fails to be transitive!
This is not a problem, because $\equiv$ can be seen as just a tool to proof things about *causal equivalence* $\equiv_{caus}$.
$N_1 \equiv_{caus} N_2$ if they have the same *processes* or *causal nets*.

$$N_1 \equiv N_2 \Rightarrow N_1 \equiv_{caus} N_2.$$

Showing agreement between an operational and denotational net semantics of CCSP

Olderog proposes a relation $\equiv$ between nets, called *strong bisimulation*.

1. $[\![P]\!]_{op} \equiv [\![P]\!]_{den}$ for any P for which both sides are defined.
2. $\equiv$ respects concurrency.
3. $\equiv$ has the typical congruence properties.

But it fails to be transitive!

This is not a problem, because $\equiv$ can be seen as just a tool to proof things about *causal equivalence* $\equiv_{caus}$.

$N_1 \equiv_{caus} N_2$ if they have the same *processes* or *causal nets*.

$$N_1 \equiv N_2 \Rightarrow N_1 \equiv_{caus} N_2.$$

$\equiv_{caus}$ respects concurrency.

Showing agreement between an operational and denotational net semantics of CCSP

$\equiv_{caus}$ is a *linear time* equivalence:

$$[\![a(b+c)]\!]_{op} \equiv_{caus} [\![ab+ac]\!].$$

This is less good for capturing phenomena like *deadlock behaviour*.

Showing agreement between an operational and denotational net semantics of CCSP

$\equiv_{caus}$ is a *linear time* equivalence:

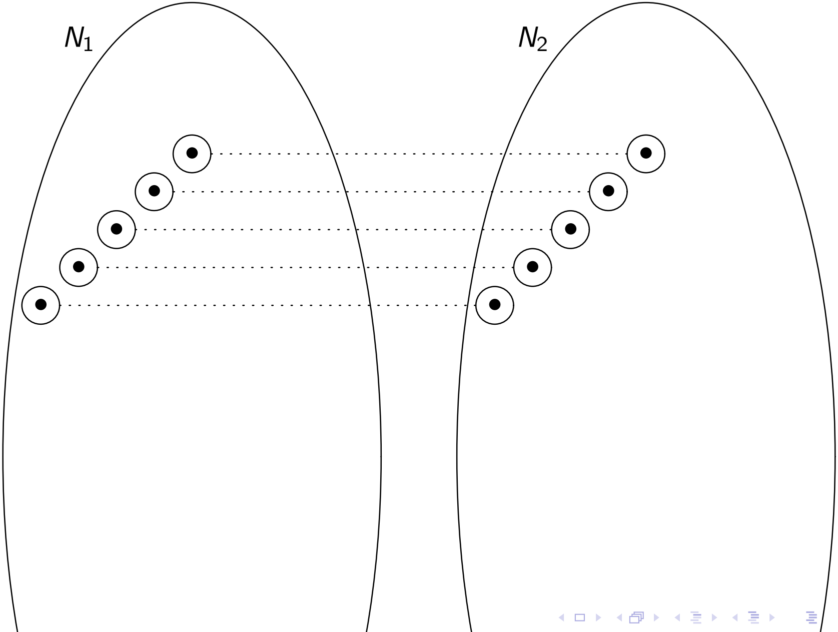
$$[\![a(b+c)]\!]_{op} \equiv_{caus} [\![ab+ac]\!].$$

This is less good for capturing phenomena like *deadlock behaviour*.

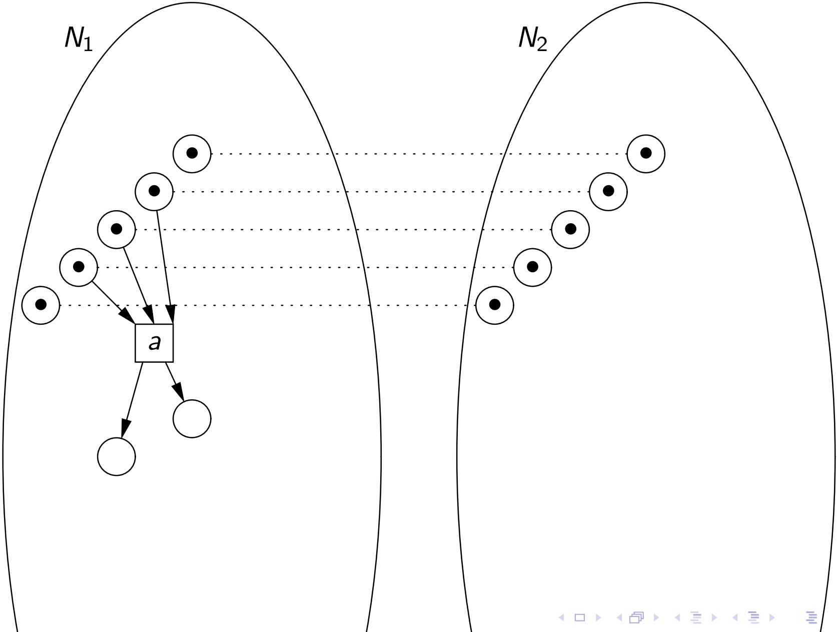
My contribution today is the proposal of a new branching time equivalence that can play the rôle of $\equiv_{caus}$.

I call it *structure preserving bisimilarity* $\underline{\leftrightarrow}_{sp}$.

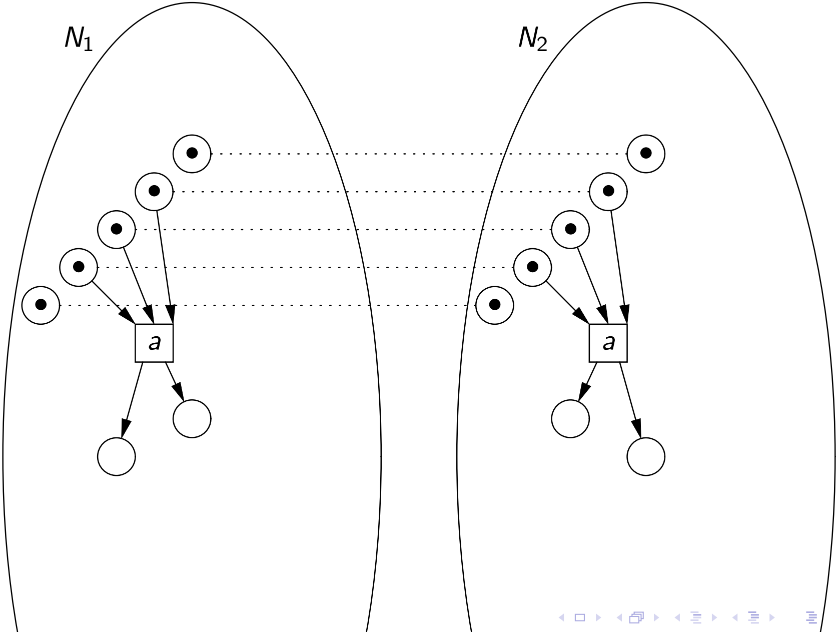
Structure Preserving Bisimulation



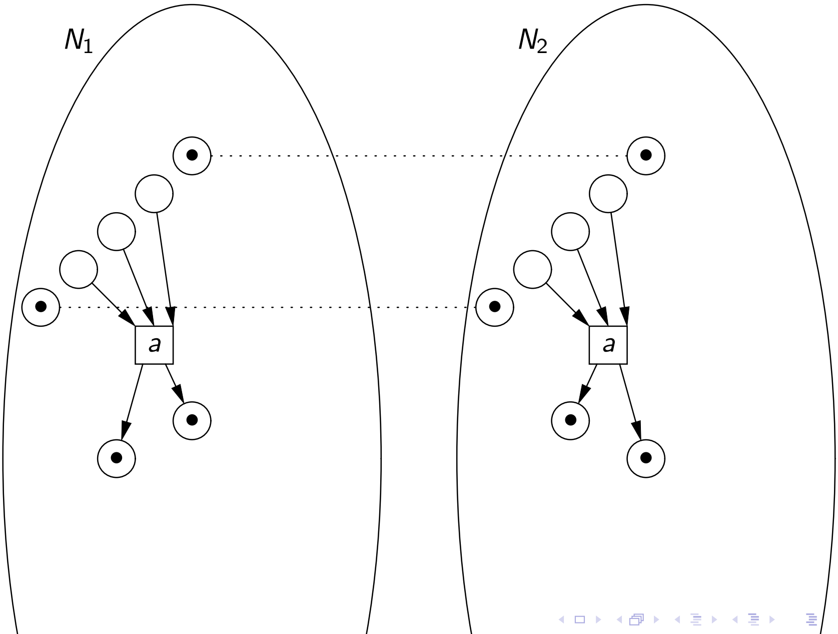
Structure Preserving Bisimulation



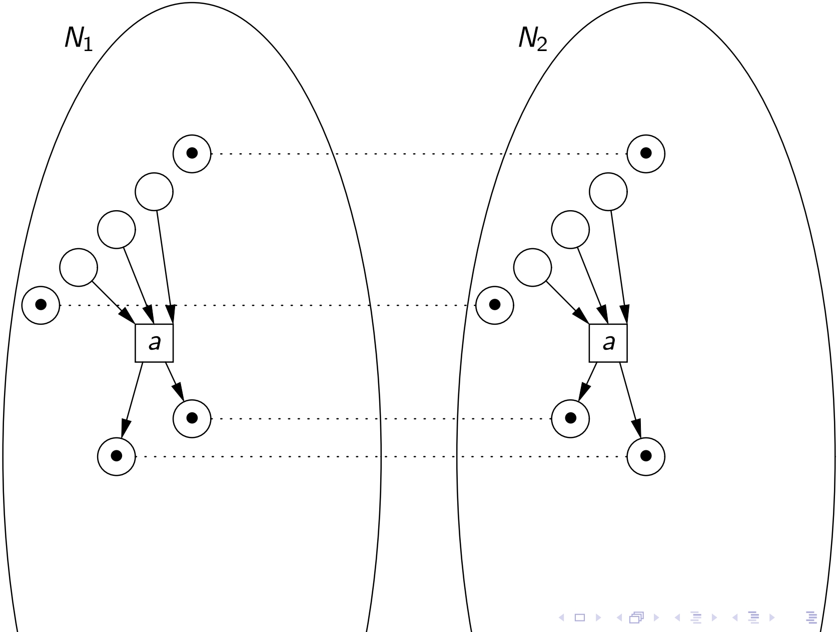
Structure Preserving Bisimulation



Structure Preserving Bisimulation



Structure Preserving Bisimulation



Criteria for choosing this equivalence

2. It should capture concurrency. $a \| b \neq ab + ba$ causality

8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality

8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]

8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each

8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1; a_2$

8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1; a_2$
6. It should be finer than $\equiv_{caus}$.

8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1 ; a_2$
6. It should be finer than $\equiv_{caus}$.
7. It should not distinguish nets whose behaviours are patently the same, i.e. when differing only in unreachable parts.
8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Semantic equivalences on Petri nets

BRANCHING TIME

↕

LINEAR TIME

ABSTRACT FROM CAUSALITY/CONCURRENCY ⟷ CAPTURE CAUSALITY/CONCURRENCY

Semantic equivalences on Petri nets

BRANCHING TIME

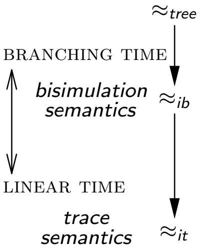
bisimulation semantics $\approx_{ib}$

LINEAR TIME

trace semantics $\approx_{it}$

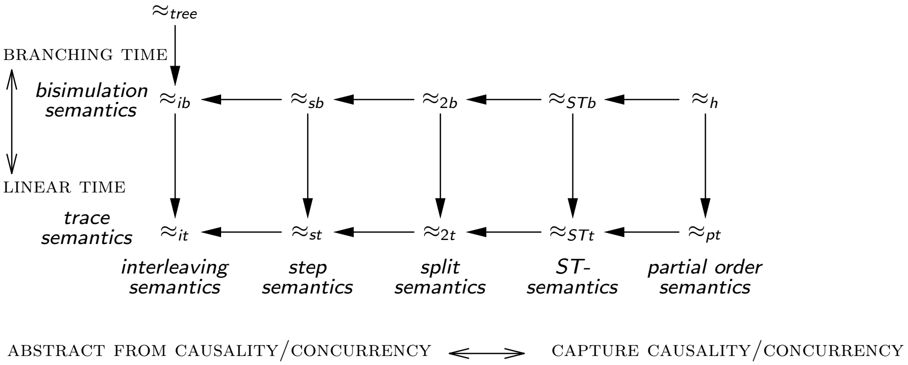
ABSTRACT FROM CAUSALITY/CONCURRENCY ⟷ CAPTURE CAUSALITY/CONCURRENCY

Semantic equivalences on Petri nets

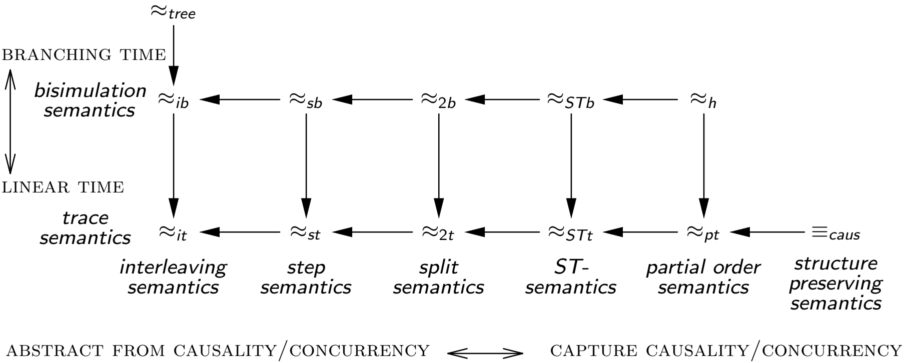


ABSTRACT FROM CAUSALITY/CONCURRENCY $\longleftrightarrow$ CAPTURE CAUSALITY/CONCURRENCY

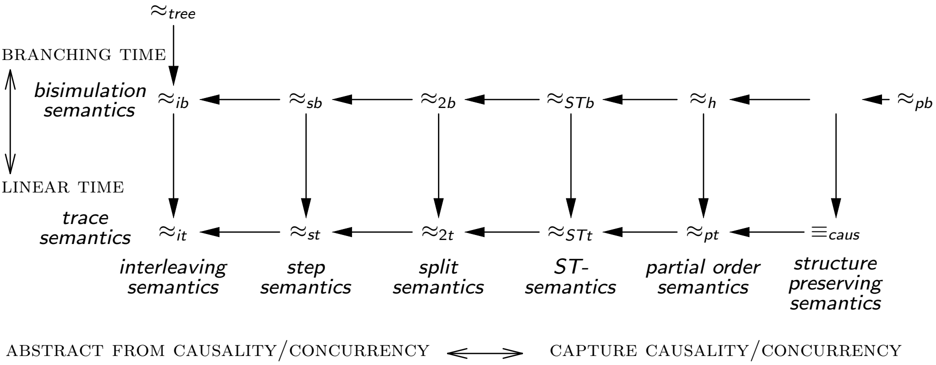
Semantic equivalences on Petri nets



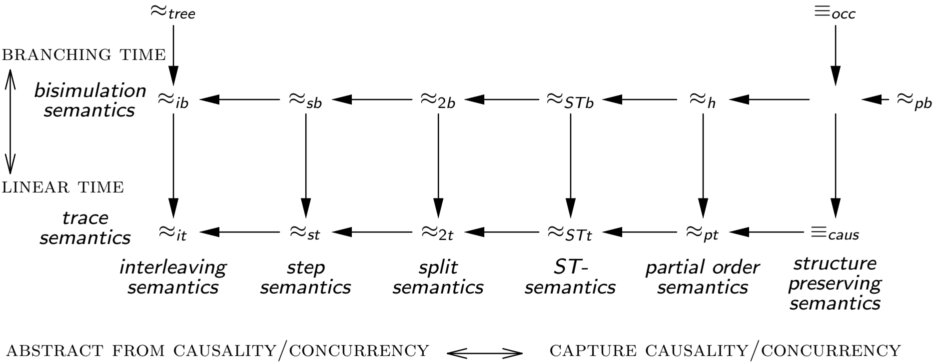
Semantic equivalences on Petri nets



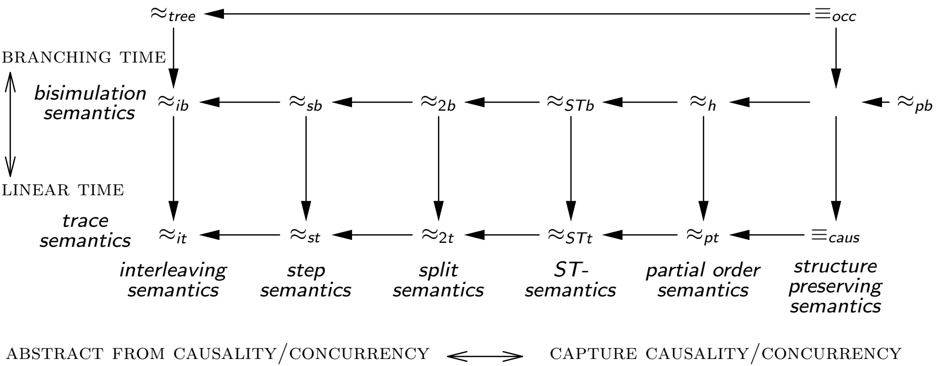
Semantic equivalences on Petri nets



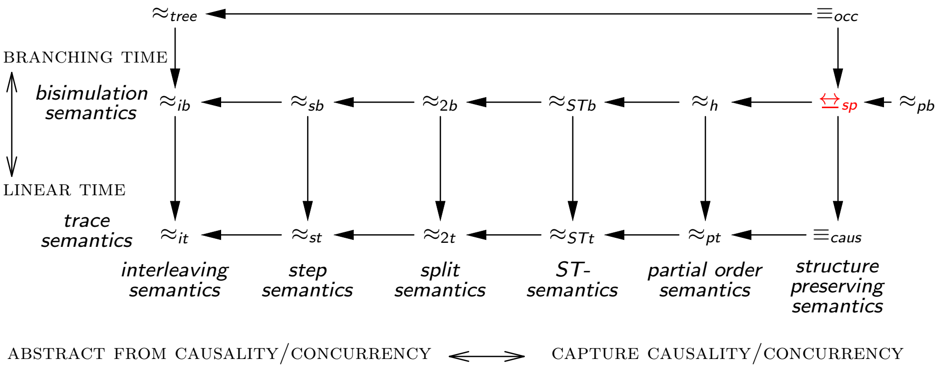
Semantic equivalences on Petri nets



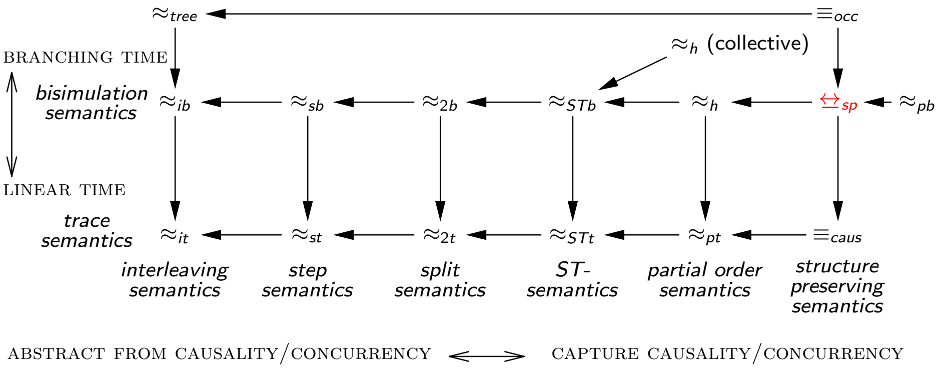
Semantic equivalences on Petri nets



Semantic equivalences on Petri nets



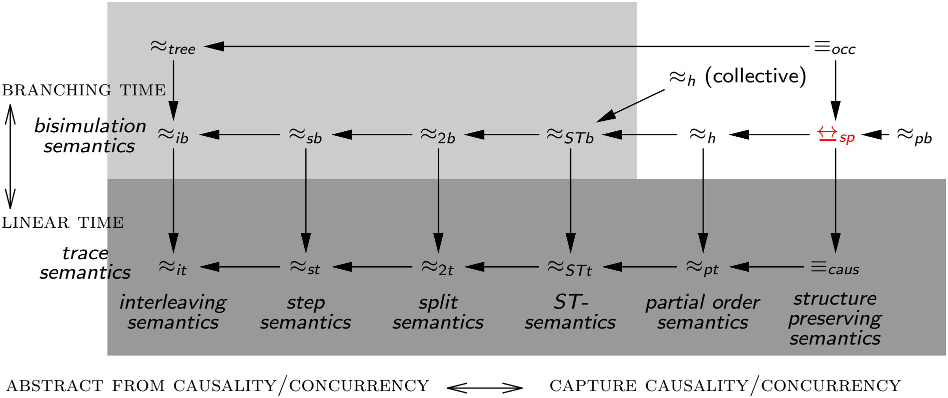
Semantic equivalences on Petri nets



Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1 ; a_2$
6. It should be finer than $\equiv_{caus}$.
7. It should not distinguish nets whose behaviours are patently the same, i.e. when differing only in unreachable parts.
8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

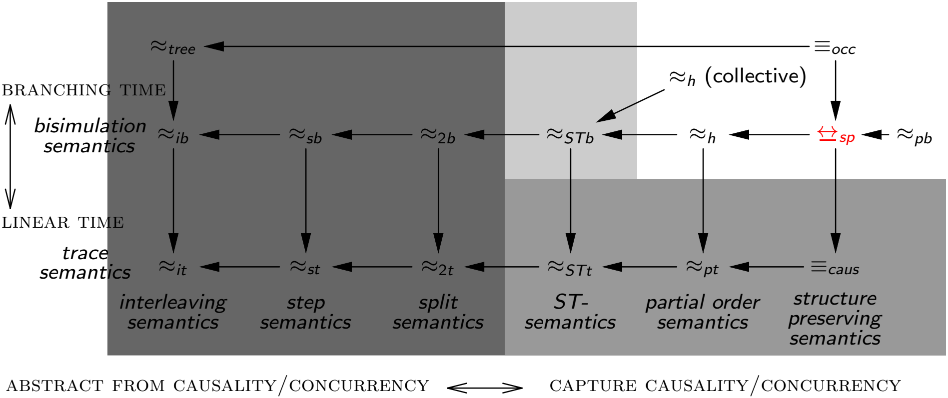
Semantic equivalences on Petri nets



Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1 ; a_2$
6. It should be finer than $\equiv_{caus}$.
7. It should not distinguish nets whose behaviours are patently the same, i.e. when differing only in unreachable parts.
8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

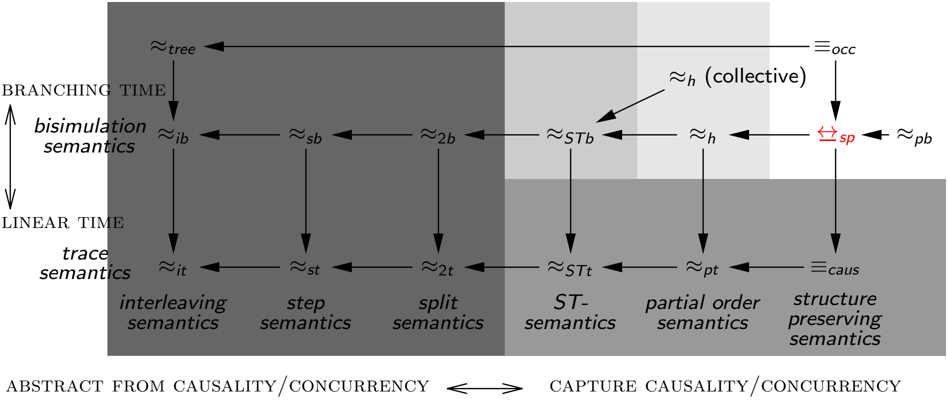
Semantic equivalences on Petri nets



Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1; a_2$
6. It should be finer than $\equiv_{caus}$.
7. It should not distinguish nets whose behaviours are patently the same, i.e. when differing only in unreachable parts.
8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

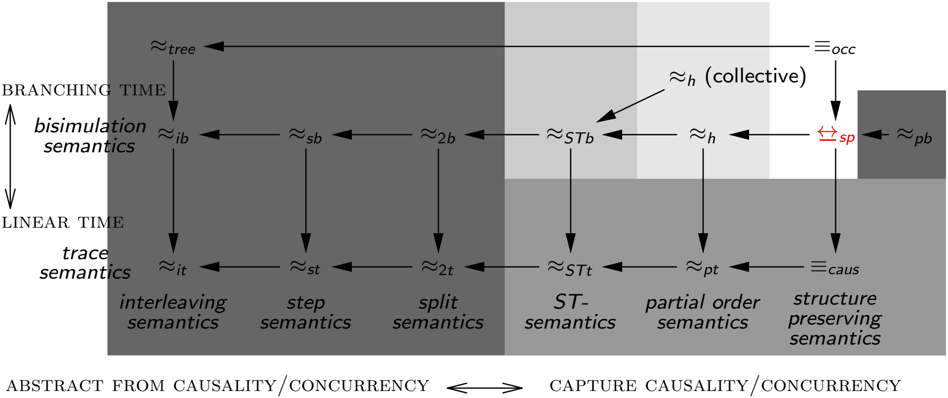
Semantic equivalences on Petri nets



Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1 ; a_2$
6. It should be finer than $\equiv_{caus}$.
7. It should not distinguish nets whose behaviours are patently the same, i.e. when differing only in unreachable parts.
8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

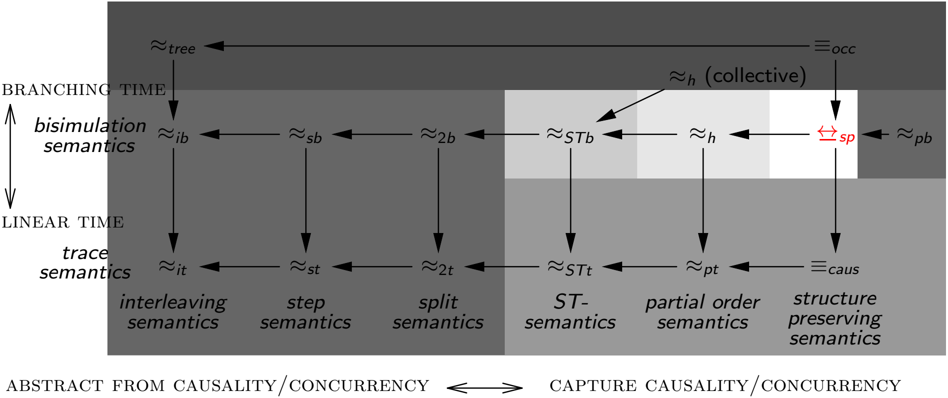
Semantic equivalences on Petri nets



Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1 ; a_2$
6. It should be finer than $\equiv_{caus}$.
7. It should not distinguish nets whose behaviours are patently the same, i.e. when differing only in unreachable parts.
8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Semantic equivalences on Petri nets



Criteria for choosing this equivalence

1. It should be a branching time equivalence.
2. It should capture concurrency. $a \| b \neq ab + ba$ causality
3. It should respect *inevitability*. [Mazurkiewicz]
4. It should be *real-time consistent*. a and b one minute each
5. It should be *preserved under action refinement*. $a \mapsto a_1 ; a_2$
6. It should be finer than $\equiv_{caus}$.
7. It should not distinguish nets whose behaviours are patently the same, i.e. when differing only in unreachable parts.
8. It should be a congruence for CCSP.
9. $[\![P]\!]_{op} \approx [\![P]\!]_{den}$ for any P for which both sides are defined.

Inevitability

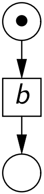
I will show you Petri nets featuring a transition b.
I will ask you by a show of hands whether you hold b to be inevitable.

Inevitability

I will show you Petri nets featuring a transition b.
I will ask you by a show of hands whether you hold b to be inevitable.

Question 0: Are you insufficiently familiar with Petri nets to answer such questions, or decline to answer for any other reason?

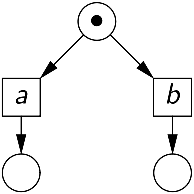
Question 1:



(CCSP expression: b)

Inevitability

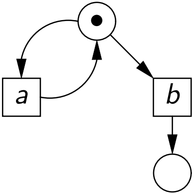
Question 2:



(CCSP expression: $a + b$)

Inevitability

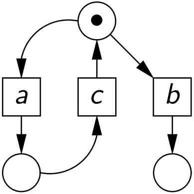
Question 3:



(CCSP expression: E with $E \stackrel{def}{=} a.E + b$.)

Inevitability

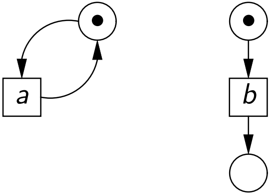
Question 4:



(CCSP expression: E' with $E' \stackrel{def}{=} a.c.E' + b$.)

Inevitability

Question 5:



(CCSP expression: $A \| b$ with $A \stackrel{def}{=} a.A$.)

Fairness

In the literature I found only 4 meaningful types of fairness assumptions:

1. Progress
2. Justness
3. Weak Fairness
4. Strong Fairness

These form a hierarchy, thus creating 5 assumptional states.

Fairness

Shop with 2 customers.
When in the shop, a customer is waiting expectantly to be served.
Upon being served, the customer leaves the shop, but usually returns right away to buy something else.

A customer may leave the shop anytime, and possibly return later.

Fairness

Shop with 2 customers.
When in the shop, a customer is waiting expectantly to be served.
Upon being served, the customer leaves the shop, but usually returns right away to buy something else.

A customer may leave the shop anytime, and possibly return later.

Failure of weak fairness: Customer A remains forever in the shop, ready to be served, but is never served, because the clerk is always busy serving customer B.

Fairness

Shop with 2 customers.

When in the shop, a customer is waiting expectantly to be served.

Upon being served, the customer leaves the shop, but usually returns right away to buy something else.

A customer may leave the shop anytime, and possibly return later.

Failure of weak fairness: Customer A remains forever in the shop, ready to be served, but is never served, because the clerk is always busy serving customer B.

Failure of strong fairness: Customer A occasionally leaves the shop, but always returns in the hope to get served. Yet, this never occurs, because the clerk is always busy serving customer B.

Fairness

Shop with 2 customers.

When in the shop, a customer is waiting expectantly to be served.

Upon being served, the customer leaves the shop, but usually returns right away to buy something else.

A customer may leave the shop anytime, and possibly return later.

Failure of weak fairness: Customer A remains forever in the shop, ready to be served, but is never served, because the clerk is always busy serving customer B.

Failure of strong fairness: Customer A occasionally leaves the shop, but always returns in the hope to get served. Yet, this never occurs, because the clerk is always busy serving customer B.

Failure of progress: Customer A remains forever in the shop, ready to be served, but no-one is ever served. The clerk stares pathetically at the customer(s) without doing anything.

Fairness

Shop with 2 customers.

When in the shop, a customer is waiting expectantly to be served.

Upon being served, the customer leaves the shop, but usually returns right away to buy something else.

A customer may leave the shop anytime, and possibly return later.

Failure of weak fairness: Customer A remains forever in the shop, ready to be served, but is never served, because the clerk is always busy serving customer B.

Failure of strong fairness: Customer A occasionally leaves the shop, but always returns in the hope to get served. Yet, this never occurs, because the clerk is always busy serving customer B.

Failure of progress: Customer A remains forever in the shop, ready to be served, but no-one is ever served. The clerk stares pathetically at the customer(s) without doing anything.

Failure of justness: There are two counters with a clerk each. Customer A is the only customer at counter 1, yet never is served, while customer B is being served repeatedly at counter 2.

Fairness in CCSP

Are the following processes guaranteed to do action b eventually?

Assuming	nothing	progr.	justness	wk. f.	str. fair.
b					
$a + b$					
E with $E \stackrel{def}{=} a.E + b$.					
E' with $E' \stackrel{def}{=} a.c.E' + b$.					
$A \| b$ with $A \stackrel{def}{=} a.A$					

Fairness in CCSP

Are the following processes guaranteed to do action b eventually?

Assuming	nothing	progr.	justness	wk. f.	str. fair.
b	–				
$a + b$	–				
E with $E \stackrel{def}{=} a.E + b$.	–				
E' with $E' \stackrel{def}{=} a.c.E' + b$.	–				
$A \| b$ with $A \stackrel{def}{=} a.A$	–				

Fairness in CCSP

Are the following processes guaranteed to do action b eventually?

Assuming	nothing	progr.	justness	wk. f.	str. fair.
b	–	✓	✓	✓	✓
$a + b$	–				
E with $E \stackrel{def}{=} a.E + b.$	–				
E' with $E' \stackrel{def}{=} a.c.E' + b.$	–				
$A \| b$ with $A \stackrel{def}{=} a.A$	–				

Fairness in CCSP

Are the following processes guaranteed to do action b eventually?

Assuming	nothing	progr.	justness	wk. f.	str. fair.
b	–	✓	✓	✓	✓
$a + b$	–	–	–	–	–
E with $E \stackrel{def}{=} a.E + b.$	–				
E' with $E' \stackrel{def}{=} a.c.E' + b.$	–				
$A \| b$ with $A \stackrel{def}{=} a.A$	–				

Fairness in CCSP

Are the following processes guaranteed to do action b eventually?

Assuming	nothing	progr.	justness	wk. f.	str. fair.
b	–	✓	✓	✓	✓
$a + b$	–	–	–	–	–
E with $E \stackrel{def}{=} a.E + b.$	–	–	–	✓	✓
E' with $E' \stackrel{def}{=} a.c.E' + b.$	–				
$A \| b$ with $A \stackrel{def}{=} a.A$	–				

Fairness in CCSP

Are the following processes guaranteed to do action b eventually?

Assuming	nothing	progr.	justness	wk. f.	str. fair.
b	–	✓	✓	✓	✓
$a+b$	–	–	–	–	–
E with $E \stackrel{def}{=} a.E + b.$	–	–	–	✓	✓
E' with $E' \stackrel{def}{=} a.c.E' + b.$	–	–	–	–	✓
$A\|\|b$ with $A \stackrel{def}{=} a.A$	–				

Fairness in CCSP

Are the following processes guaranteed to do action b eventually?

Assuming	nothing	progr.	justness	wk. f.	str. fair.
b	–	✓	✓	✓	✓
$a + b$	–	–	–	–	–
E with $E \stackrel{def}{=} a.E + b.$	–	–	–	✓	✓
E' with $E' \stackrel{def}{=} a.c.E' + b.$	–	–	–	–	✓
$A \| b$ with $A \stackrel{def}{=} a.A$	–	–	✓	✓	✓

Fairness in process algebra

Strong or weak fairness can be

- indispensable for certain applications, such as a correctness proof of the alternating bit protocol.
- patently wrong when used where not appropriate.

E with $E \stackrel{def}{=} a.E + b.0$.

Fairness in process algebra

Strong or weak fairness can be

- indispensable for certain applications, such as a correctness proof of the alternating bit protocol.
- patently wrong when used where not appropriate.

E with $E \stackrel{def}{=} a.E + b.0$.

- could be a spec. of a mobile phone
  - b is a successful dialling attempt
  - a is an unsuccessful dialling attempt.

  Fairness amounts to saying that if you try often enough, dialling will succeed.
  This is wishful thinking.
  The real world is not fair.

Fairness in process algebra

Strong or weak fairness can be

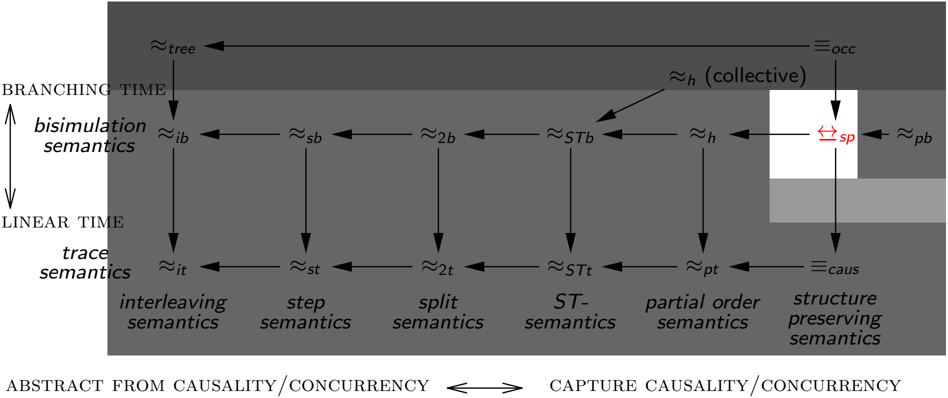
- indispensable for certain applications, such as a correctness proof of the alternating bit protocol.
- patently wrong when used where not appropriate.

E with $E \stackrel{def}{=} a.E + b.0$.

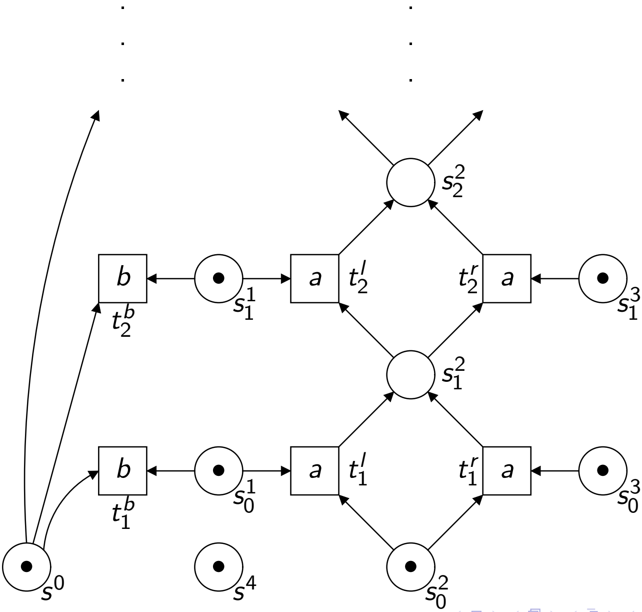
- could be a spec. of a mobile phone
  - b is a successful dialling attempt
  - a is an unsuccessful dialling attempt.

  Fairness amounts to saying that if you try often enough, dialling will succeed.
  This is wishful thinking.
  The real world is not fair.
- When assuming strong or weak fairness, we loose the ability to finitely specify a system like E above that *does* allow an infinite sequences of *a*s without a b.

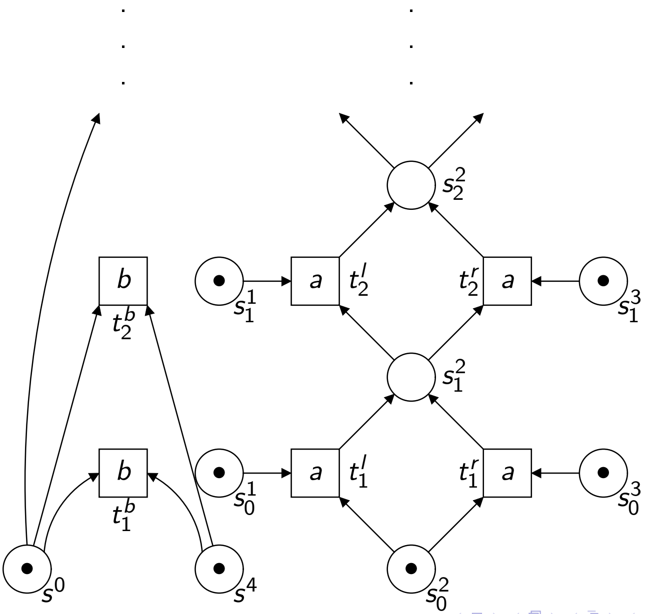
Semantic equivalences on Petri nets



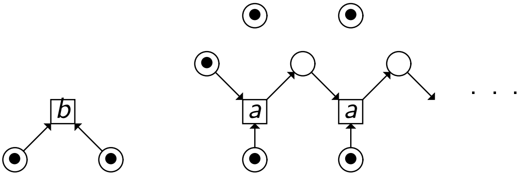
HP bisimilarity does not respect inevitability



HP bisimilarity does not respect inevitability



Causal equivalence does not respect inevitability



The causal nets of both systems are the infinite one above, and all its finite prefixes.

Conclusion

1. I proposed 9 requirements on semantic equivalences on Petri nets.
2. None of the existing equivalences satisfies all (or almost all) of these requirements.
3. I propose a new equivalence that does.
4. A major motivation of this equivalence is its suitability in establishing agreement between the denotational and operational Petri net semantics of CCSP.