Just Testing

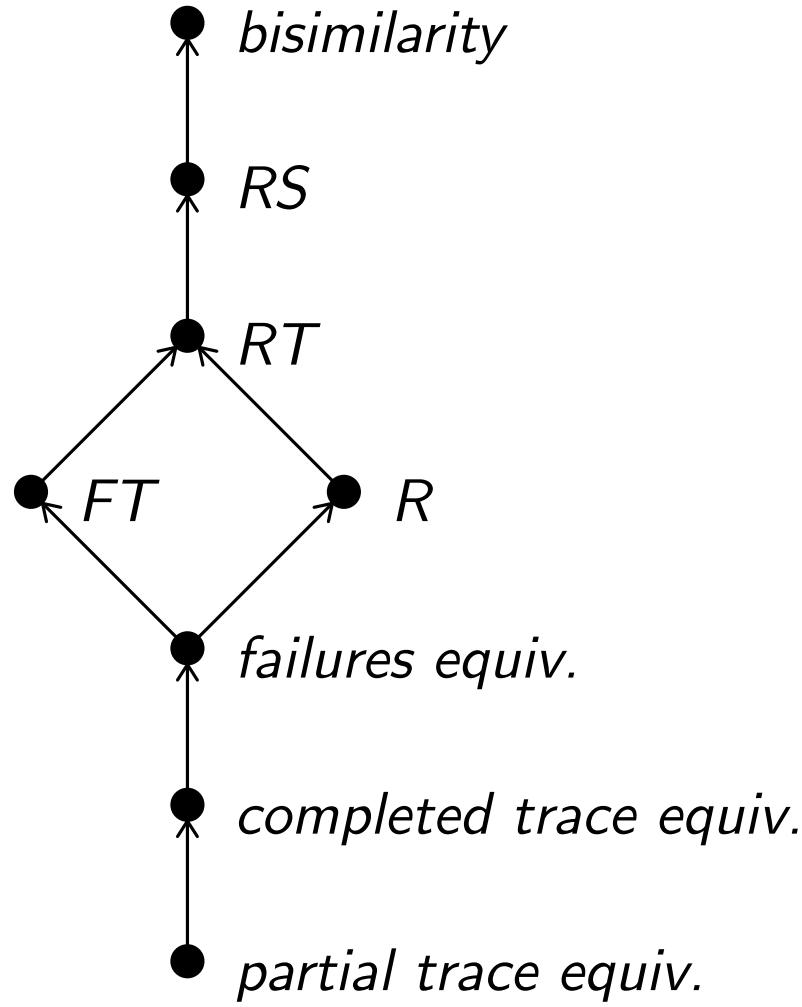
Rob van Glabbeek

University of Edinburgh, Scotland

27 April 2023

Semantic equivalences

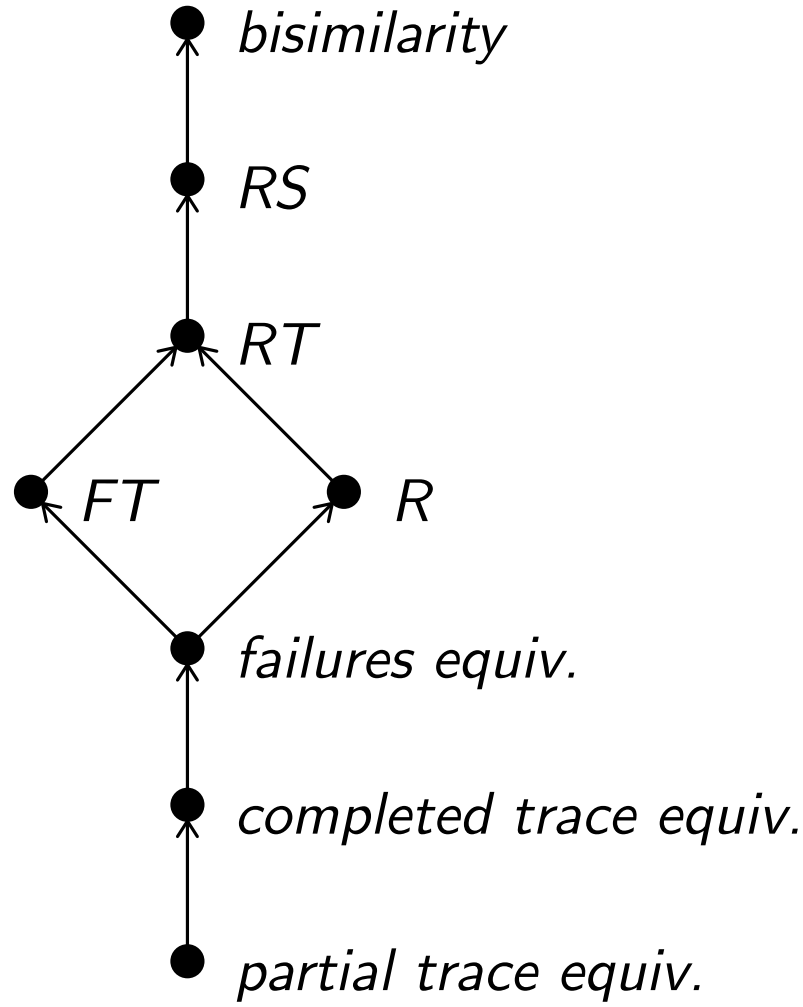
This talk is about *behavioural equivalence relations* on processes.



Semantic equivalences

This talk is about *behavioural equivalence relations* on processes.

$P \sim Q$ means that
P can safely be replaced by Q
in an appropriate context.



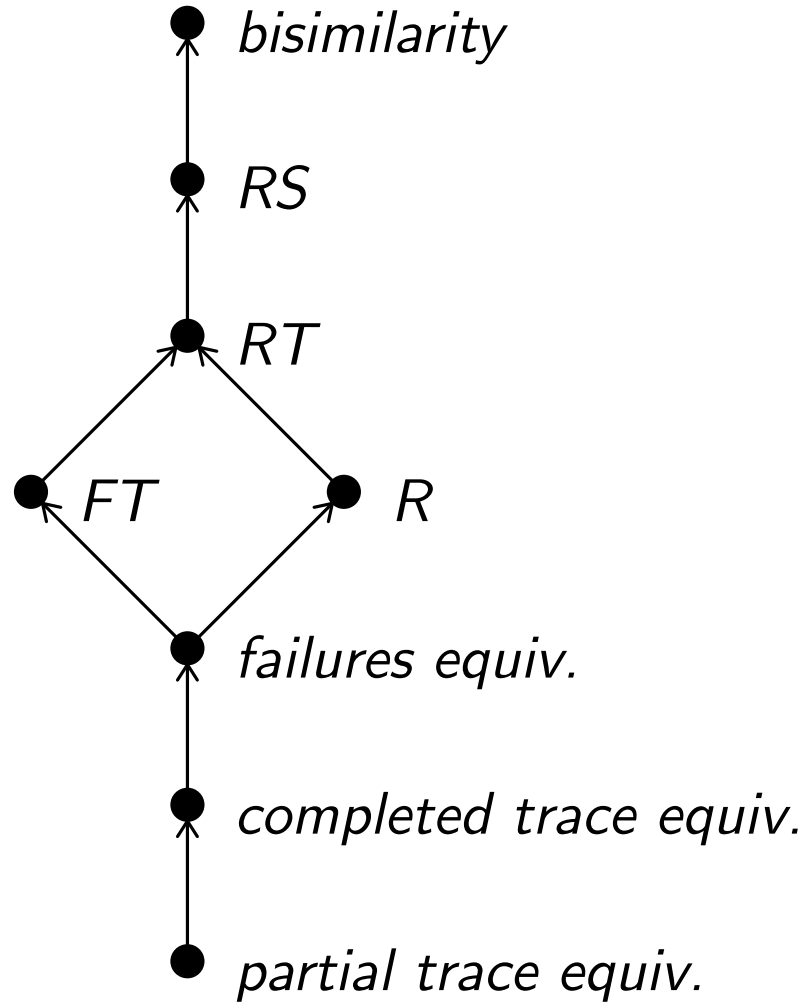
Semantic equivalences

This talk is about *behavioural equivalence relations* on processes.

$P \sim Q$ means that
P can safely be replaced by Q
in an appropriate context.

De Nicola & Hennessy:

$P \not\sim Q$ if and only if
they react differently on certain tests.



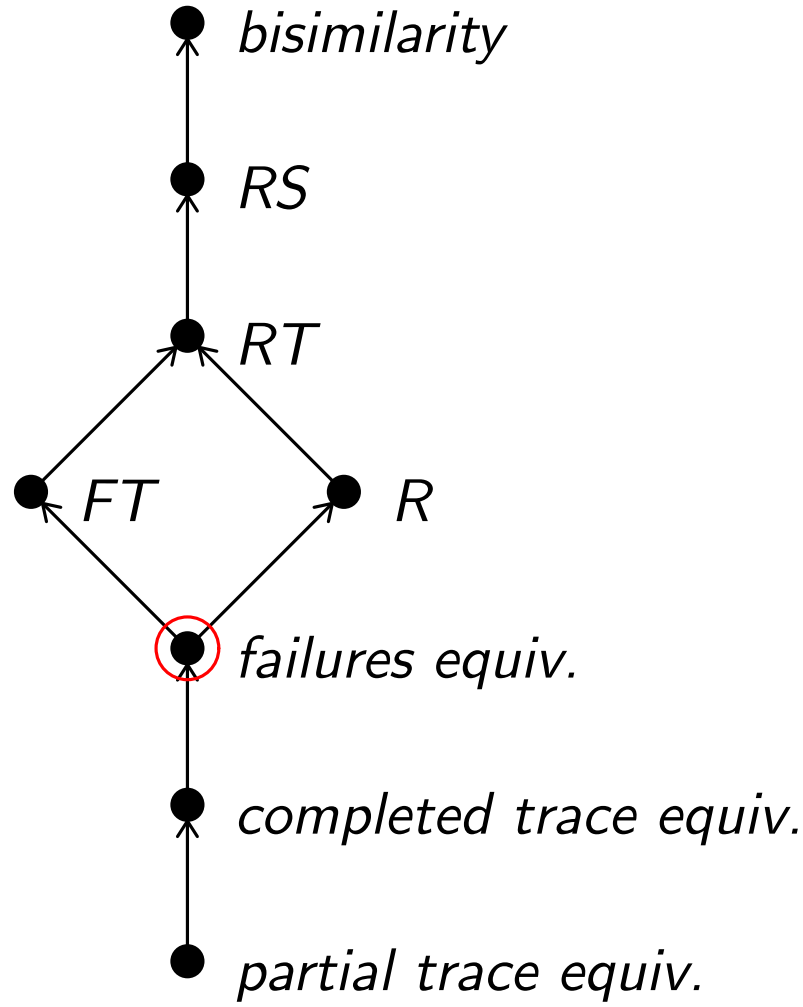
Semantic equivalences

This talk is about *behavioural equivalence relations* on processes.

$P \sim Q$ means that
P can safely be replaced by Q
in an appropriate context.

De Nicola & Hennessy:

$P \not\sim Q$ if and only if
they react differently on certain tests.



The theory of testing [DH84]

$P \not\sim Q$ if and only if they react differently on certain tests.

The theory of testing [DH84]

$P \not\sim Q$ if and only if they react differently on certain tests.

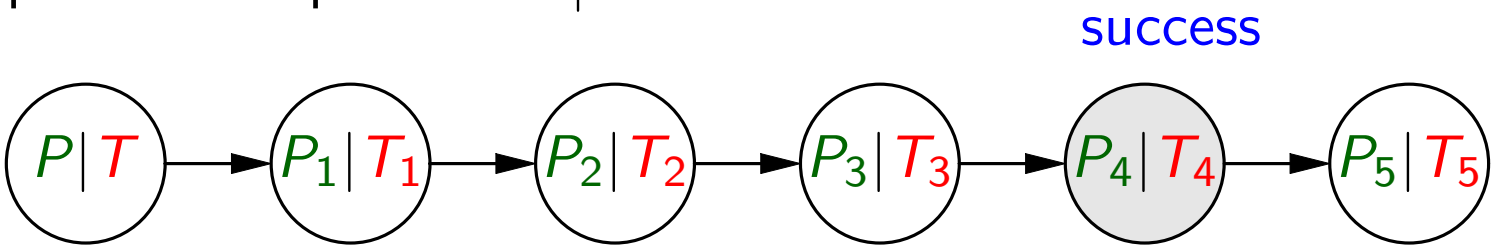
A test T if a normal process (like the processes P, Q being tested) but with a notion of *success state*.

The theory of testing [DH84]

$P \not\sim Q$ if and only if they react differently on certain tests.

A test T if a normal process (like the processes P, Q being tested) but with a notion of *success state*.

Process P *must* pass test T if each complete execution path of the parallel composition $P|T$ is successful.

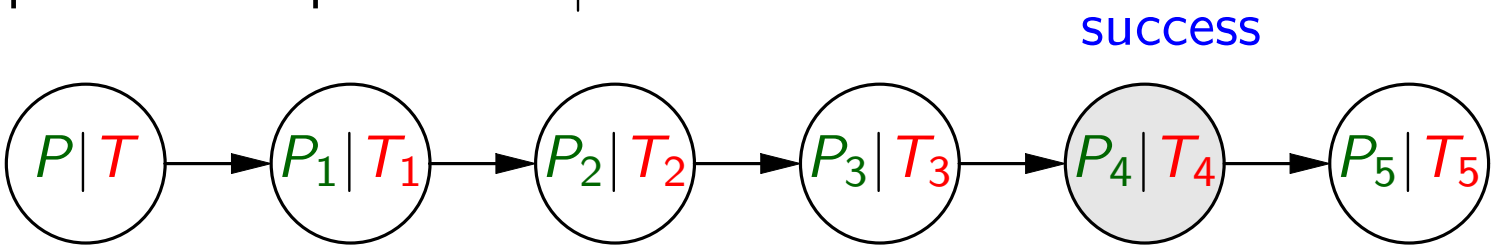


The theory of testing [DH84]

$P \not\sim Q$ if and only if they react differently on certain tests.

A test T if a normal process (like the processes P, Q being tested) but with a notion of *success state*.

Process P *must* pass test T if each complete execution path of the parallel composition $P|T$ is successful.



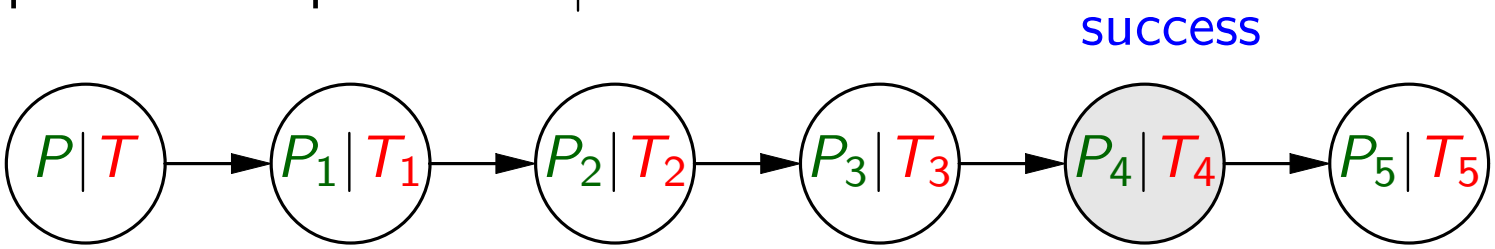
Write $P \sqsubseteq_{\text{must}} Q$ iff Q must pass any test that P must pass.
Write $P \equiv_{\text{must}} Q$ if $P \sqsubseteq_{\text{must}} Q$ and $Q \sqsubseteq_{\text{must}} Q$.

The theory of testing [DH84]

$P \not\sim Q$ if and only if they react differently on certain tests.

A test T if a normal process (like the processes P, Q being tested) but with a notion of *success state*.

Process P *must* pass test T if each complete execution path of the parallel composition $P|T$ is successful.



Write $P \sqsubseteq_{\text{must}} Q$ iff Q must pass any test that P must pass.
Write $P \equiv_{\text{must}} Q$ if $P \sqsubseteq_{\text{must}} Q$ and $Q \sqsubseteq_{\text{must}} Q$.

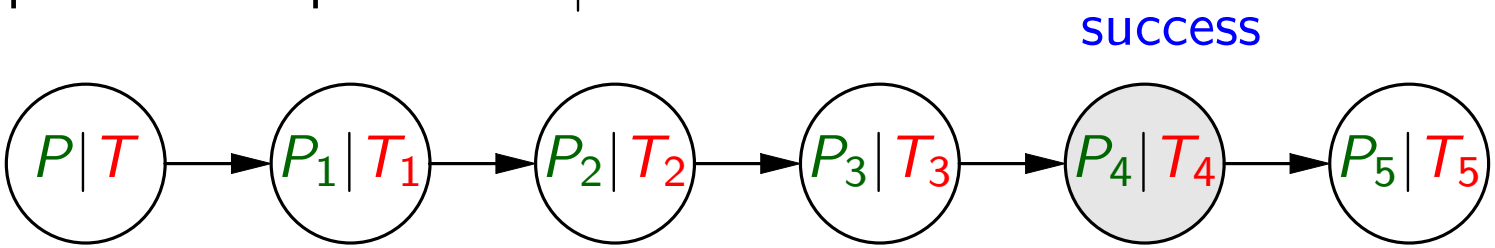
Process P *may* pass test T if some execution path of the parallel composition $P|T$ is successful.

The theory of testing [DH84]

$P \not\sim Q$ if and only if they react differently on certain tests.

A test T if a normal process (like the processes P, Q being tested) but with a notion of *success state*.

Process P *must* pass test T if each complete execution path of the parallel composition $P|T$ is successful.



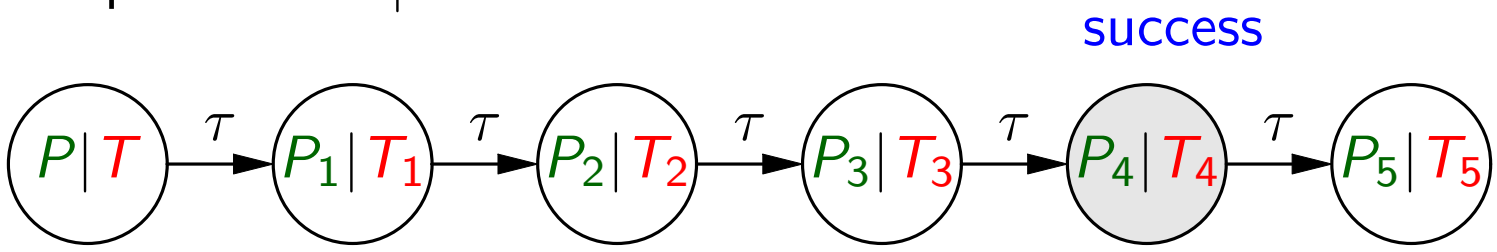
Write $P \sqsubseteq_{\text{must}} Q$ iff Q must pass any test that P must pass.
Write $P \equiv_{\text{must}} Q$ if $P \sqsubseteq_{\text{must}} Q$ and $Q \sqsubseteq_{\text{must}} Q$.

Process P *may* pass test T if some execution path of the parallel composition $P|T$ is successful.

Write $P \sqsubseteq_{\text{may}} Q$ iff Q may pass any test that P may pass.
Write $P \equiv_{\text{may}} Q$ if $P \sqsubseteq_{\text{may}} Q$ and $Q \sqsubseteq_{\text{may}} Q$.

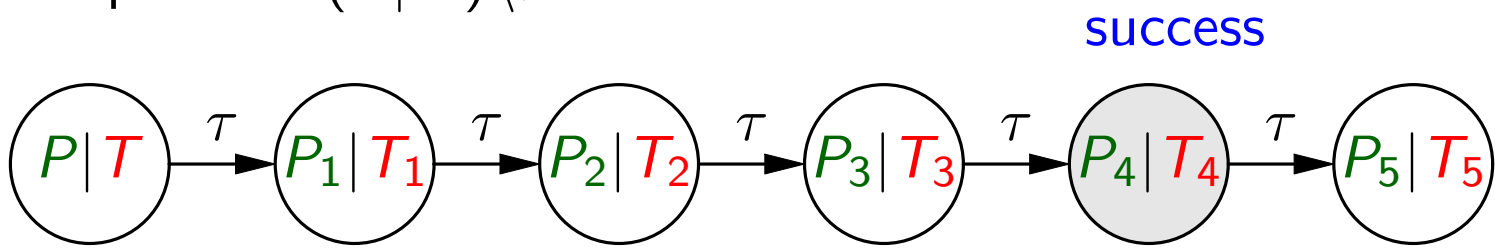
CCS versus CSP parallel composition in may-/must-testing

Process P *must* pass test T if each complete execution path of the parallel composition $P|T$ is successful.



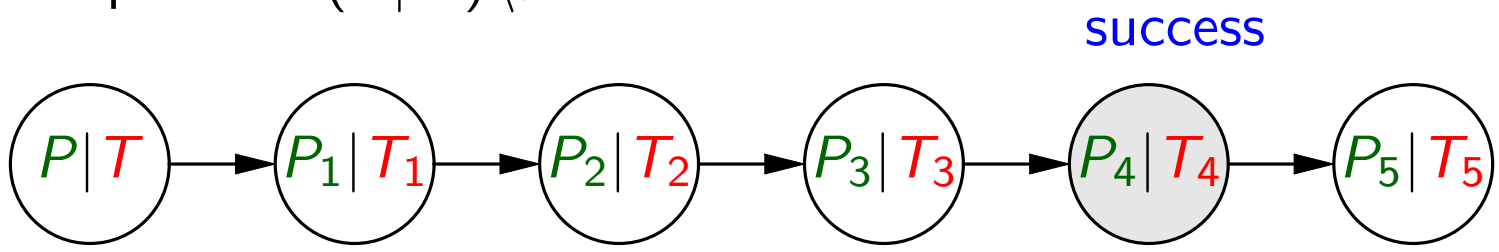
CCS versus CSP parallel composition in may-/must-testing

Process P *must* pass test T if each complete execution path of the parallel composition $(P|\,T)\backslash\mathcal{A}$ is successful.



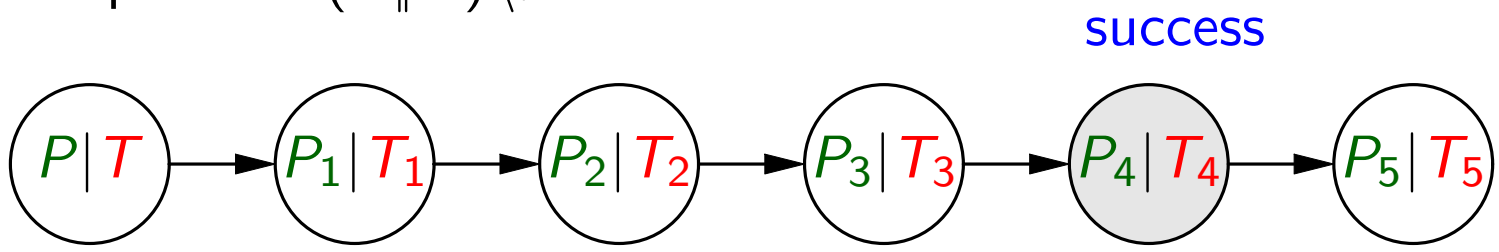
CCS versus CSP parallel composition in may-/must-testing

Process P *must* pass test T if each complete execution path of the parallel composition $(P|\,T)\backslash\mathcal{A}$ is successful.



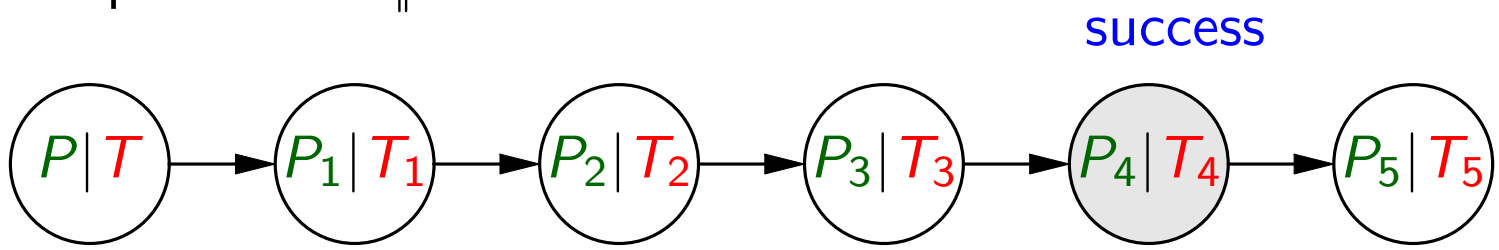
CCS versus CSP parallel composition in may-/must-testing

Process P *must* pass test T if each complete execution path of the parallel composition $(P \| T) \backslash \mathcal{A}$ is successful.



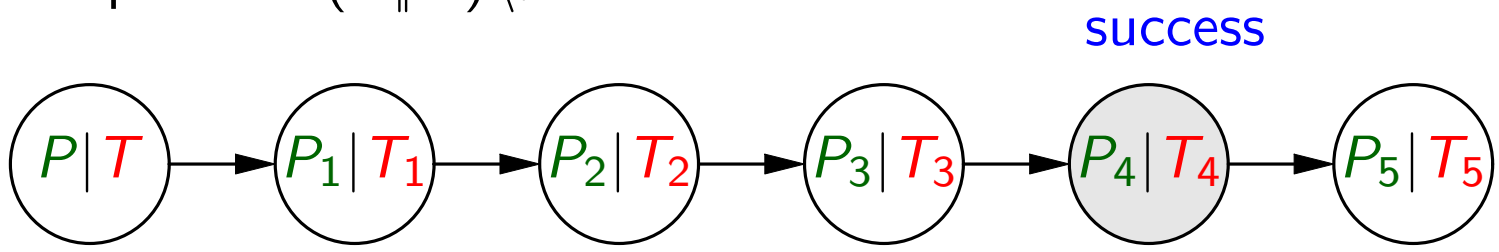
CCS versus CSP parallel composition in may-/must-testing

Process P *must* pass test T if each complete execution path of the parallel composition $P \| T$ is successful.



CCS versus CSP parallel composition in may-/must-testing

Process P *must* pass test T if each complete execution path of the parallel composition $(P \| T) \backslash \mathcal{A}$ is successful.



Completeness criteria

Does each complete execution path of a given process reach a success state?

Completeness criteria

Does each complete execution path of a given process reach a success state?

What does it mean for a path to be complete?

Completeness criteria

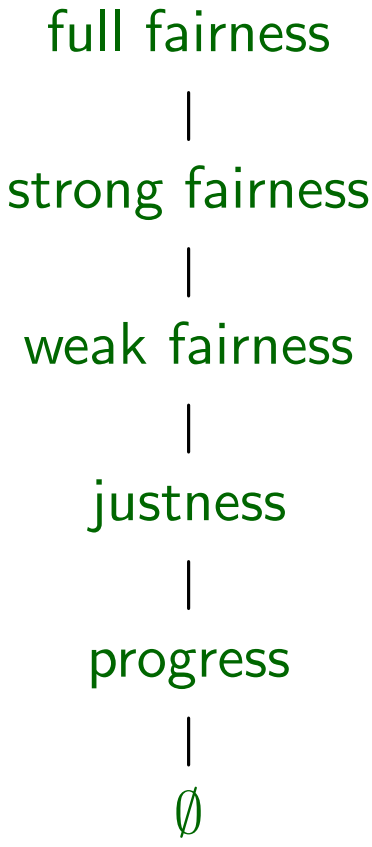
Does each complete execution path of a given process reach a success state?

What does it mean for a path to be complete?
(Assume for simplicity: only τ-transitions.)

Completeness criteria

Does each complete execution path of a given process reach a success state?

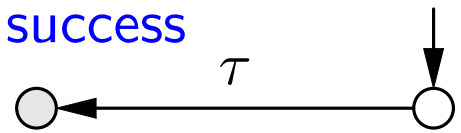
What does it mean for a path to be complete? (Assume for simplicity: only τ-transitions.)



Completeness criteria

Does each complete execution path of a given process reach a success state?

What does it mean for a path to be complete?
(Assume for simplicity: only τ-transitions.)



full fairness

|

strong fairness

|

weak fairness

|

justness

|

progress

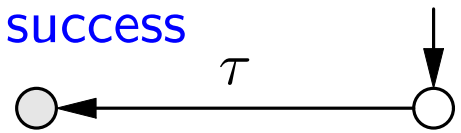
|

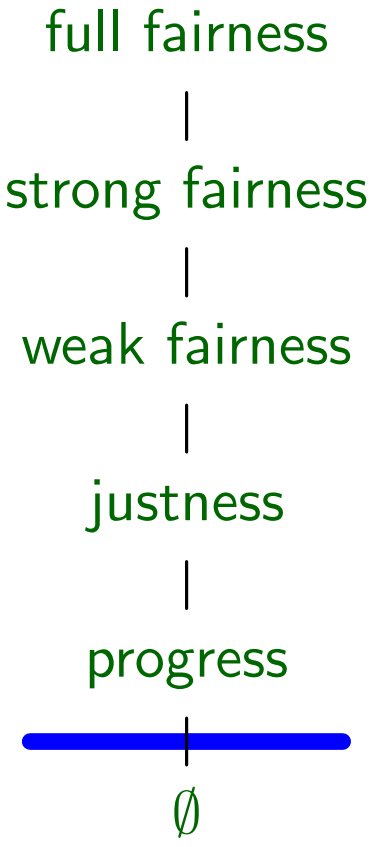
$\emptyset$

Completeness criteria

Does each complete execution path of a given process reach a success state?

What does it mean for a path to be complete?
(Assume for simplicity: only τ-transitions.)

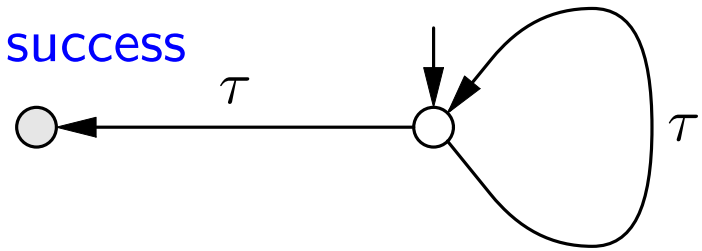


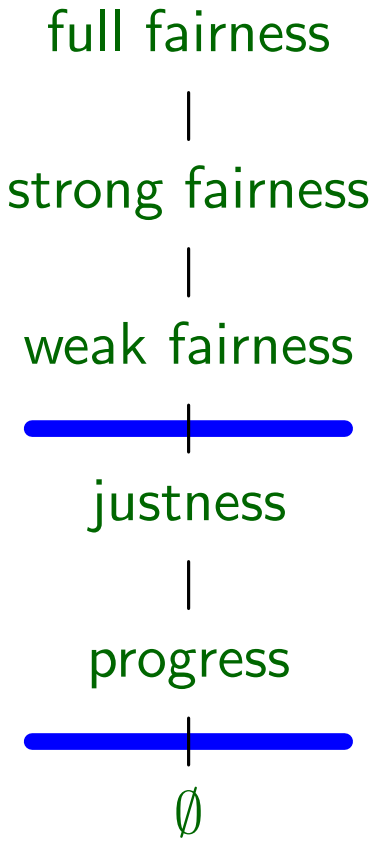


Completeness criteria

Does each complete execution path of a given process reach a success state?

What does it mean for a path to be complete? (Assume for simplicity: only τ-transitions.)

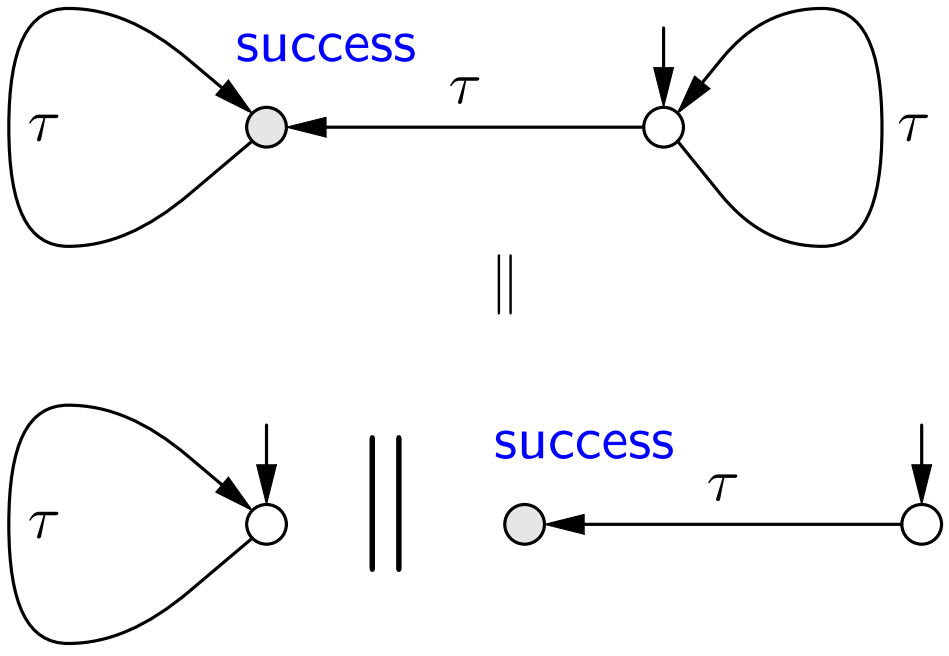


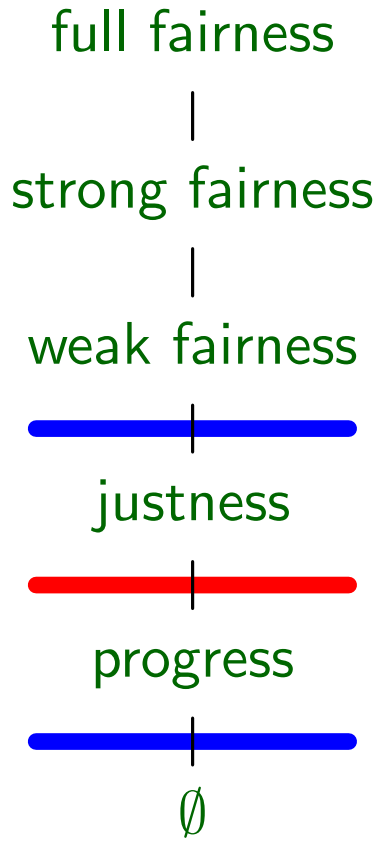


Completeness criteria

Does each complete execution path of a given process reach a success state?

What does it mean for a path to be complete? (Assume for simplicity: only τ-transitions.)

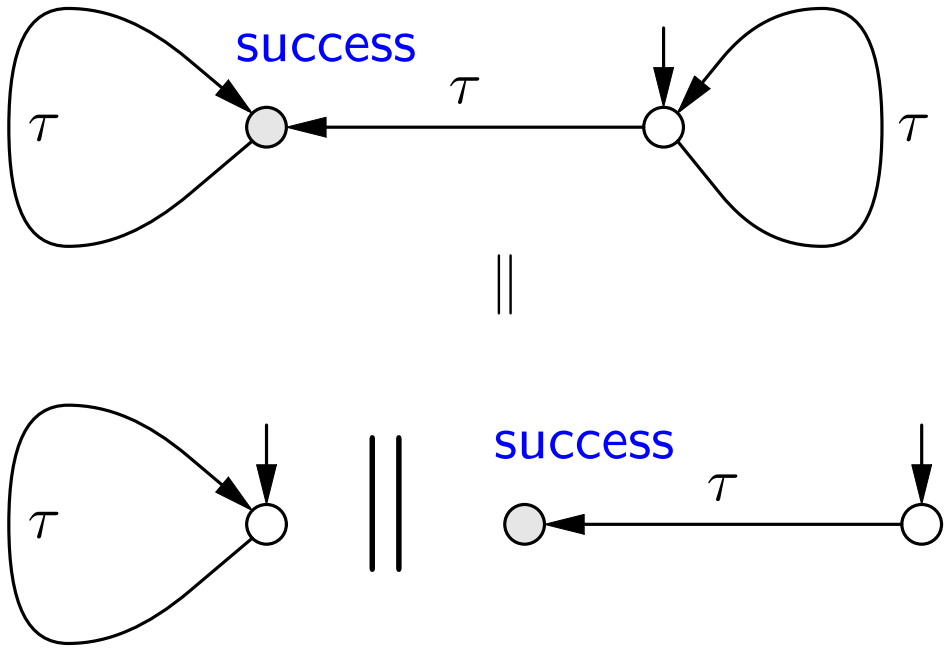


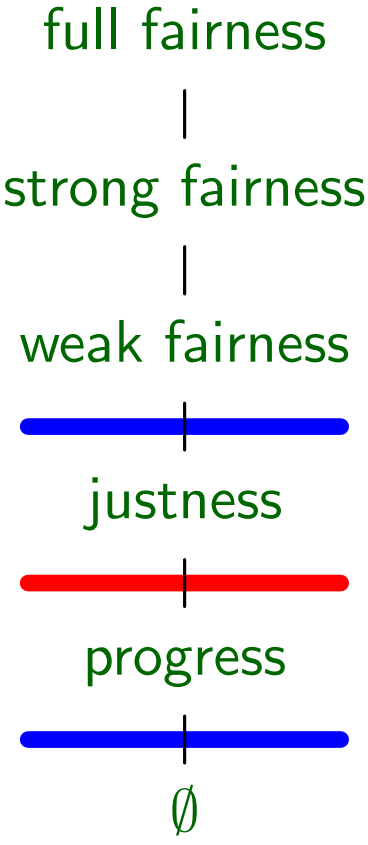


Completeness criteria

Process P *must* pass test T if each complete execution path of $P|T$ is successful.

$P \sqsubseteq_{\text{must}} Q$ iff Q passes any test that P does.
Write $P \equiv_{\text{must}} Q$ if $P \sqsubseteq_{\text{must}} Q$ and $Q \sqsubseteq_{\text{must}} Q$.

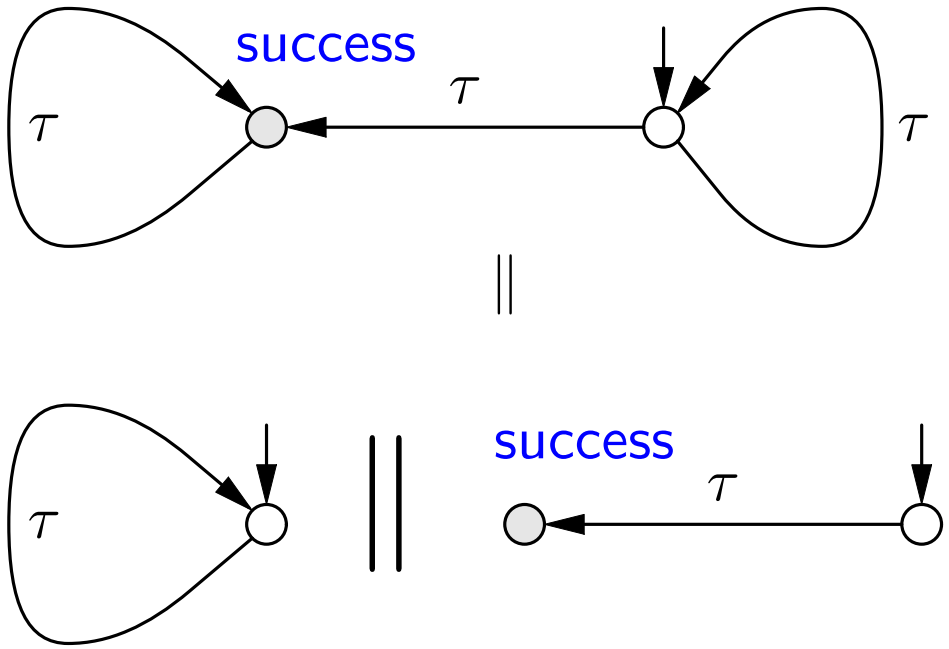


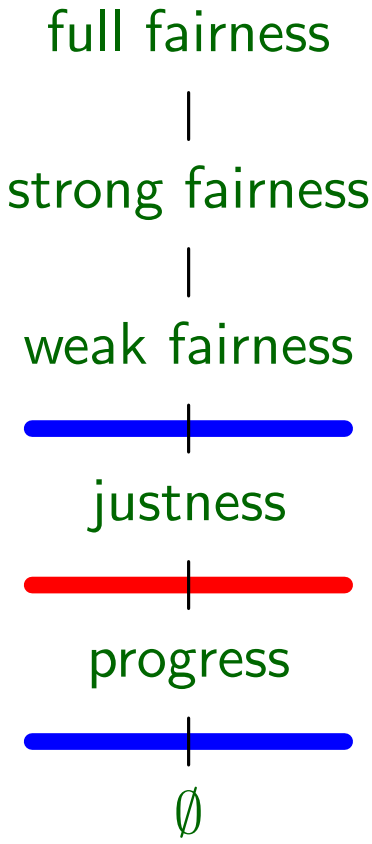


Completeness criteria

Process P *must* pass test T if each ~~complete~~ execution path of $P|T$ is successful.

$P \sqsubseteq_{\text{must}} Q$ iff Q passes any test that P does.
Write $P \equiv_{\text{must}} Q$ if $P \sqsubseteq_{\text{must}} Q$ and $Q \sqsubseteq_{\text{must}} Q$.

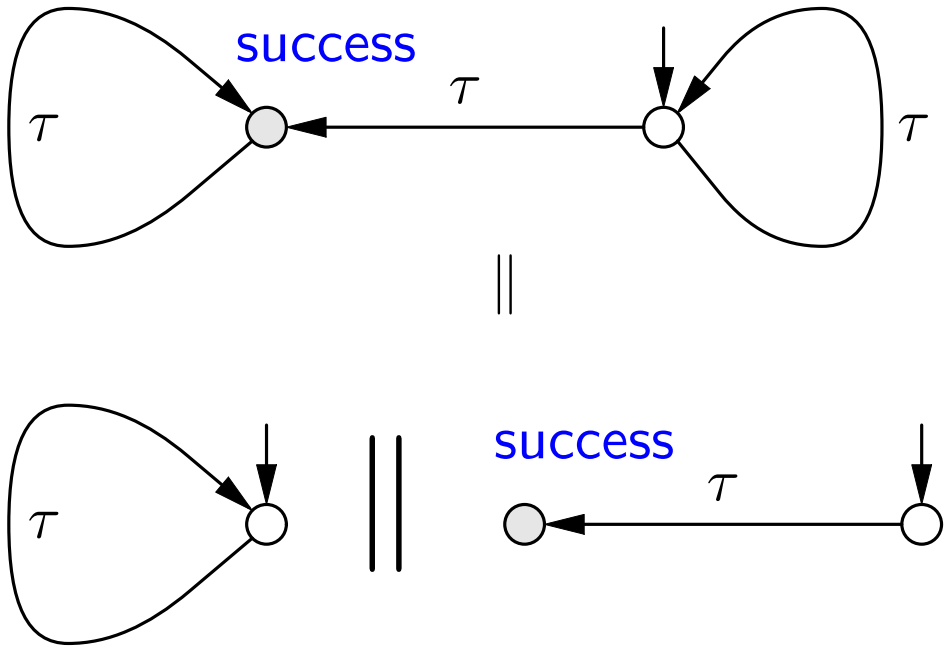


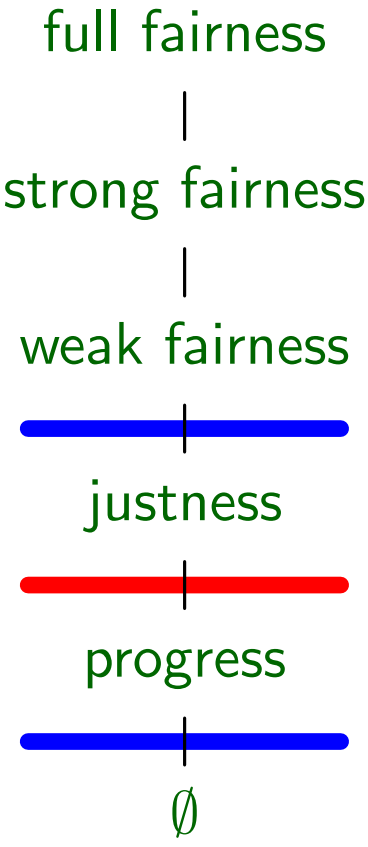


Completeness criteria

Process P *must* pass test T if each progressing execution path of $P|T$ is successful.

$P \sqsubseteq_{\text{must}} Q$ iff Q passes any test that P does.
Write $P \equiv_{\text{must}} Q$ if $P \sqsubseteq_{\text{must}} Q$ and $Q \sqsubseteq_{\text{must}} Q$.

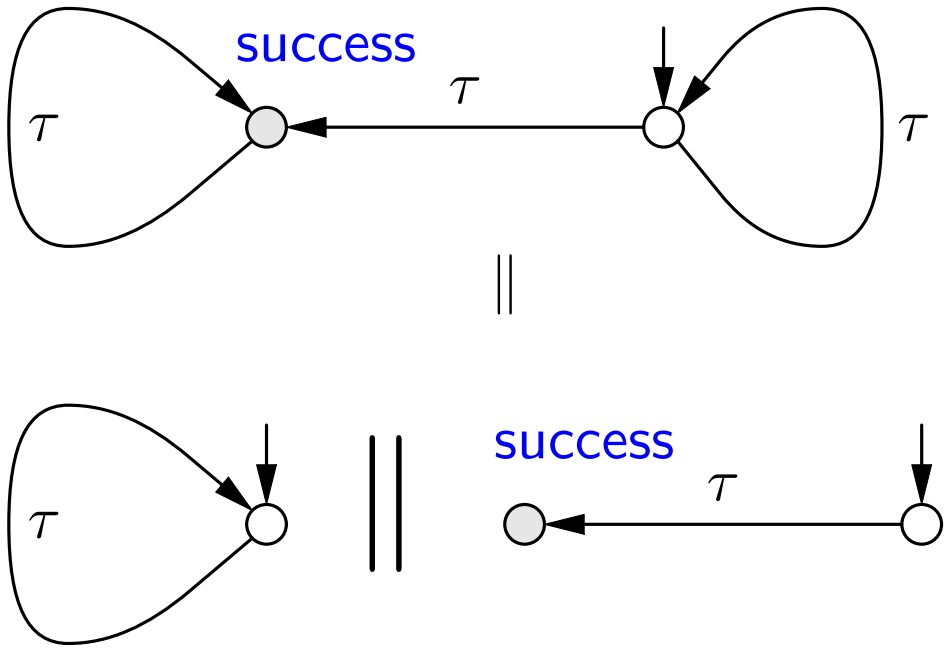


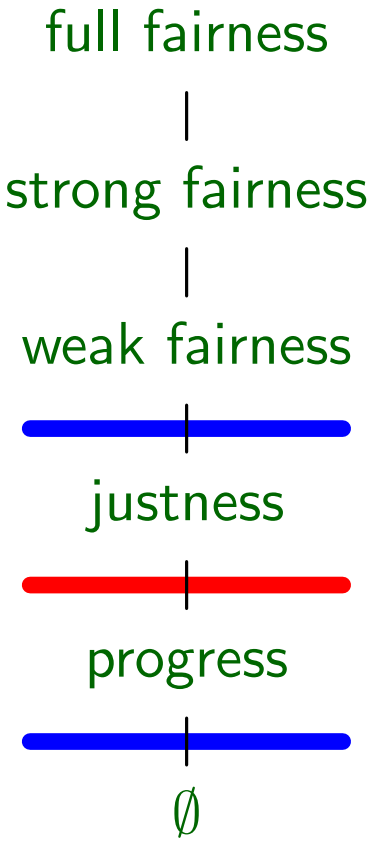


Completeness criteria

Process P *must* pass test T if each just execution path of $P|T$ is successful.

$P \sqsubseteq^{J}_{\text{must}} Q$ iff Q passes any test that P does.
Write $P \equiv^{J}_{\text{must}} Q$ if $P \sqsubseteq^{J}_{\text{must}} Q$ and $Q \sqsubseteq^{J}_{\text{must}} Q$.

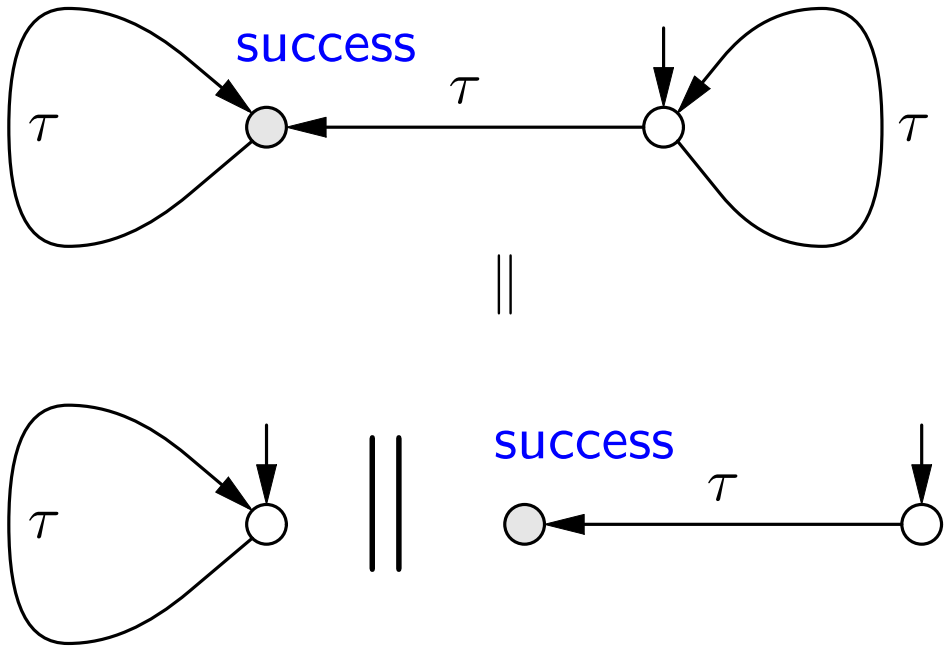


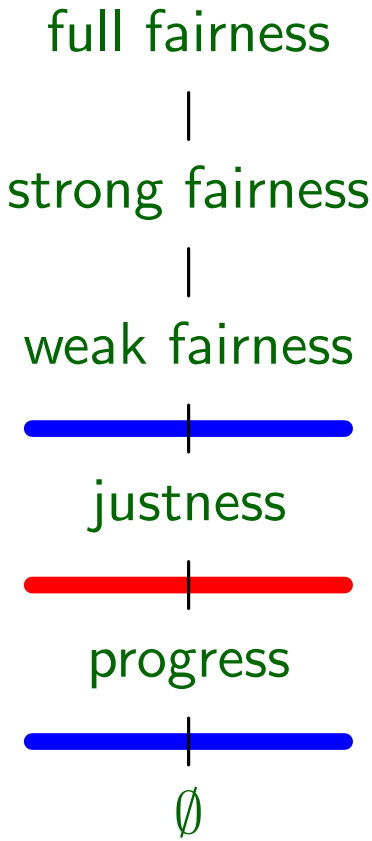


Completeness criteria

Process P *must* pass test T if each weakly fair execution path of $P|T$ is successful.

$P \sqsubseteq^{WF}_{\text{must}} Q$ iff Q passes any test that P does.
Write $P \equiv^{WF}_{\text{must}} Q$ if $P \sqsubseteq^{WF}_{\text{must}} Q$ and $Q \sqsubseteq^{WF}_{\text{must}} Q$.

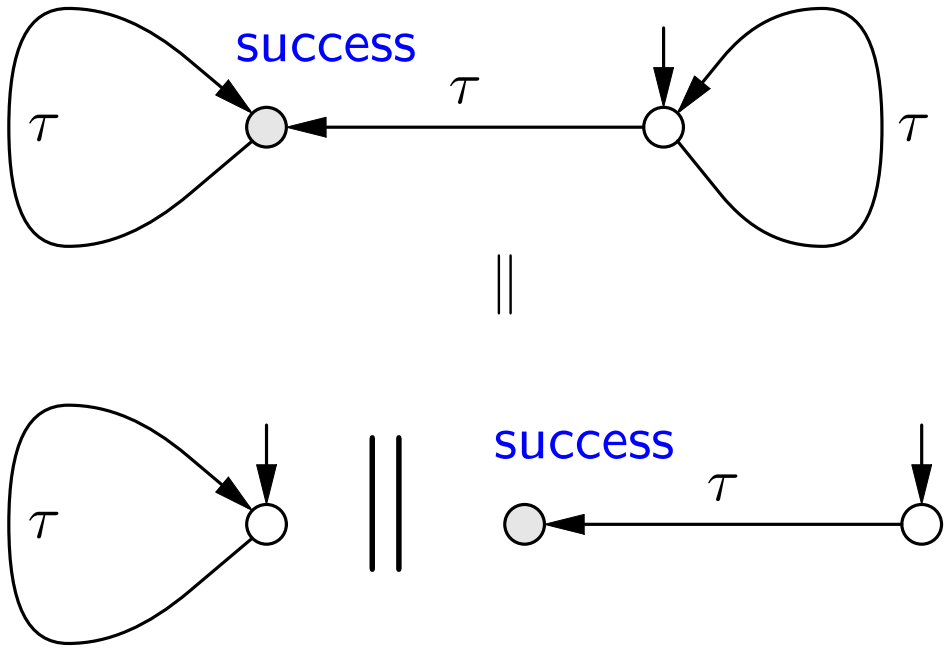


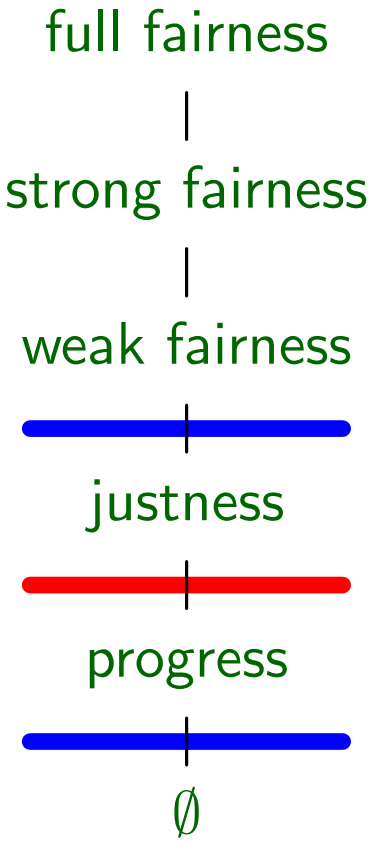


Completeness criteria

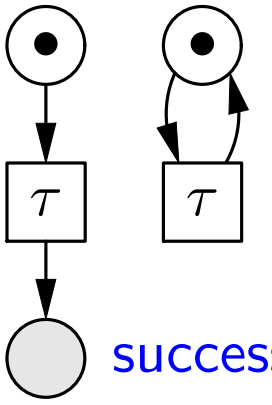
Process P *must* pass test T if each just execution path of $P|T$ is successful.

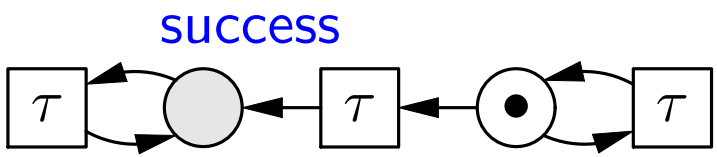
$P \sqsubseteq^{J}_{\text{must}} Q$ iff Q passes any test that P does.
Write $P \equiv^{J}_{\text{must}} Q$ if $P \sqsubseteq^{J}_{\text{must}} Q$ and $Q \sqsubseteq^{J}_{\text{must}} Q$.

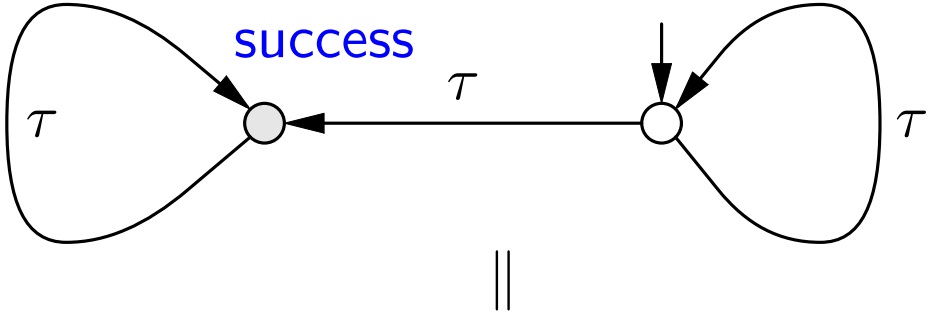


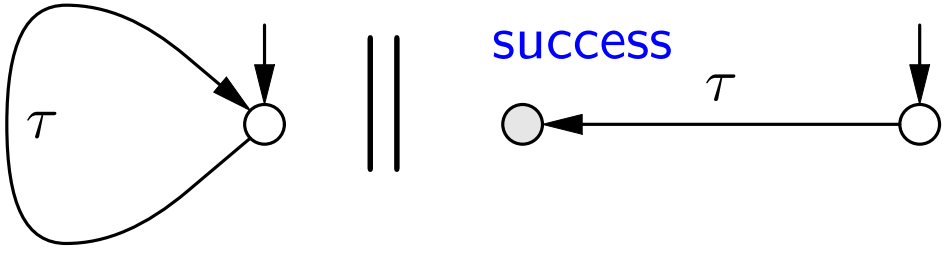


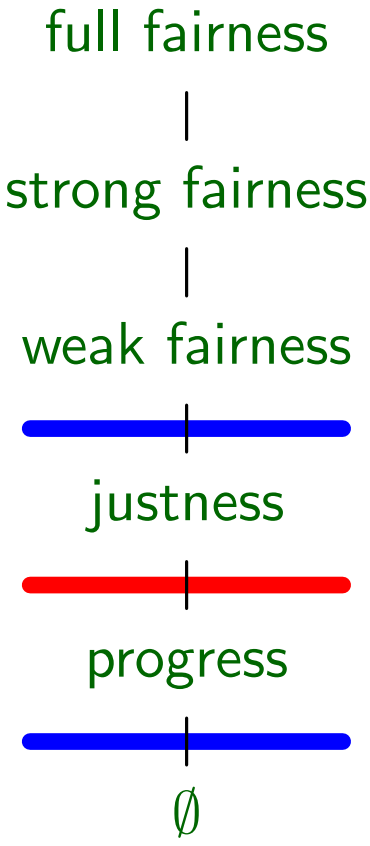
Completeness criteria



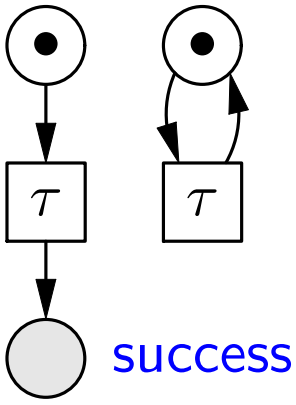


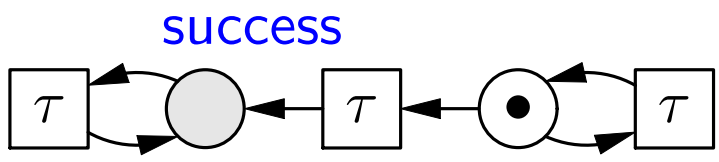






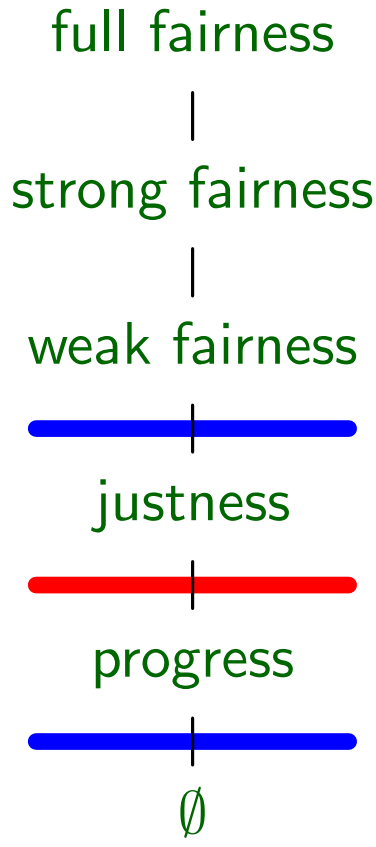
Completeness criteria





Justness on Petri nets:

Whenever a transition is enabled,
either it fires eventually
or one of its resources will be consumed.



Just must-testing

This defines just must-testing equivalence $\eqsim^{J}_{\mathrm{must}}$.

Just must-testing

This defines just must-testing equivalence $\eqq^{J}_{\mathrm{must}}$.

Main results:

Just must-testing

This defines just must-testing equivalence $\simeq^{J}_{\mathrm{must}}$.

Main results:

$\simeq^{J}_{\mathrm{must}}$ is a congruence for all operators of CSP.

Just must-testing

This defines just must-testing equivalence $\equiv^{J}_{\mathrm{must}}$.

Main results:

$\equiv^{J}_{\mathrm{must}}$ is a congruence for all operators of CSP.

$\equiv^{J}_{\mathrm{must}}$ preserves all linear time properties,
when taking justness as the underlying completeness criterion.

Just must-testing

This defines just must-testing equivalence $\simeq^{J}_{\mathrm{must}}$.

Main results:

$\simeq^{J}_{\mathrm{must}}$ is a congruence for all operators of CSP.

$\simeq^{J}_{\mathrm{must}}$ preserves all linear time properties,
when taking justness as the underlying completeness criterion.

$\simeq^{J}_{\mathrm{must}}$ is the coarsest congruence (for parallel composition,
concealment and injective relabelling) that preserves LT properties.

Just must-testing

This defines just must-testing equivalence $\equiv^{J}_{\mathrm{must}}$.

Main results:

$\equiv^{J}_{\mathrm{must}}$ is a congruence for all operators of CSP.

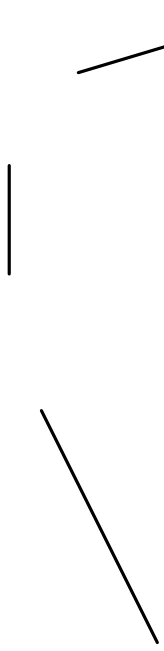
$\equiv^{J}_{\mathrm{must}}$ preserves all linear time properties,
when taking justness as the underlying completeness criterion.

$\equiv^{J}_{\mathrm{must}}$ is the coarsest congruence (for parallel composition, concealment and injective relabelling) that preserves LT properties.

$\equiv^{J}_{\mathrm{must}}$ coincides with the *fair failure equivalence* defined on Petri nets by Walter Vogler in 2002. However, Vogler did not obtain this equivalence through a (must-)testing scenario.

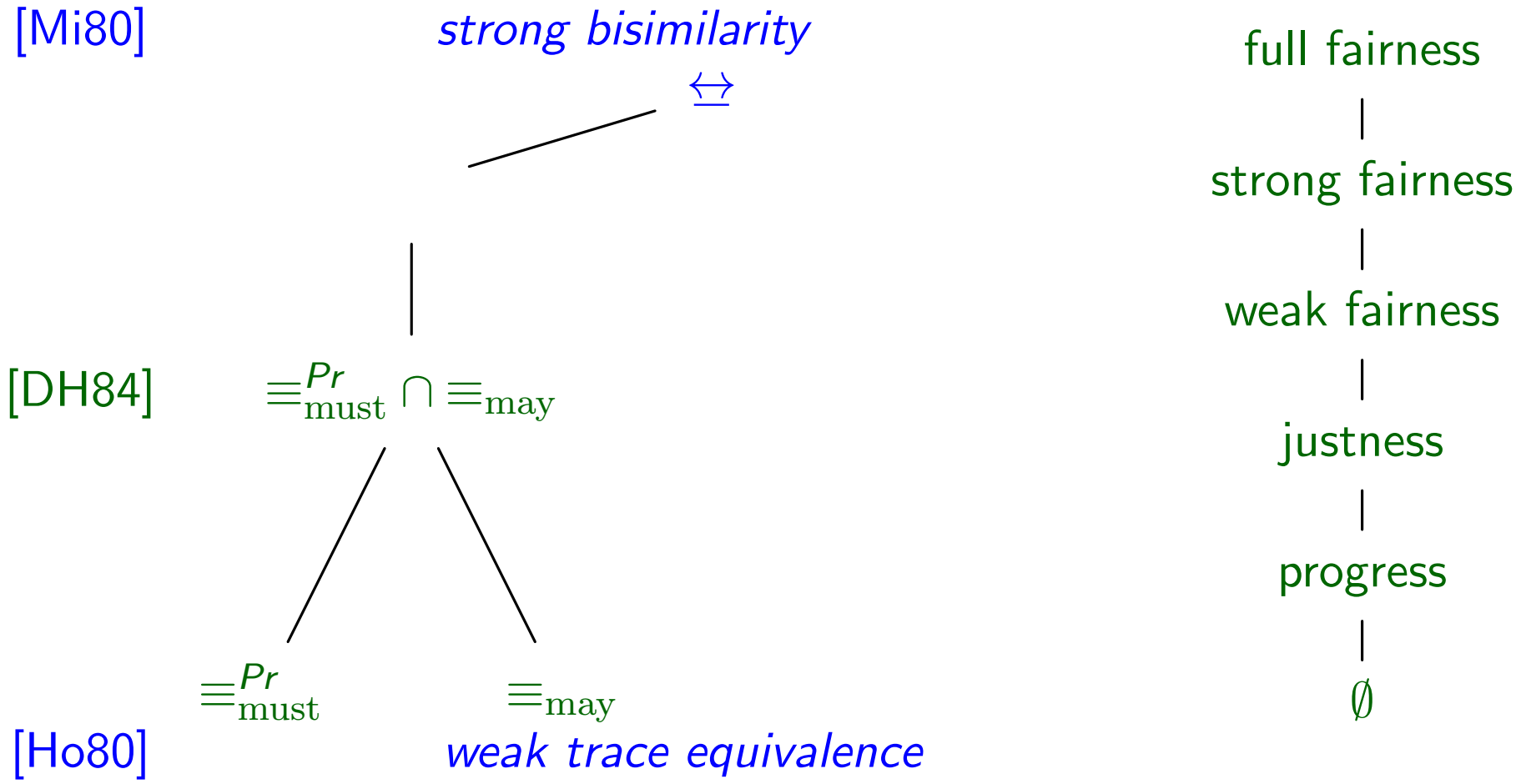
A spectrum of testing preorders and bisimilarities

[Mi80] *strong bisimilarity* $\leftrightarrows$

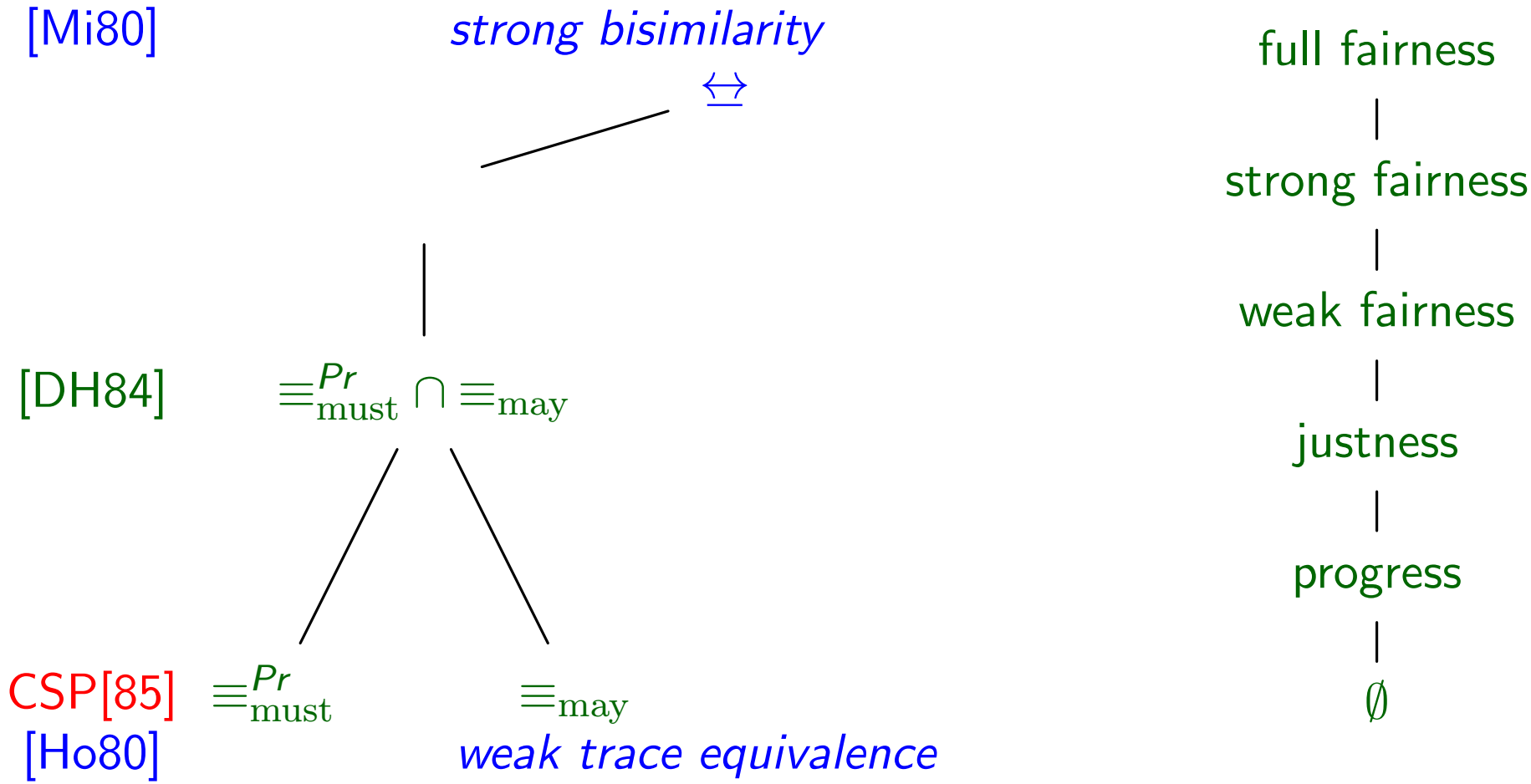


[Ho80] *weak trace equivalence*

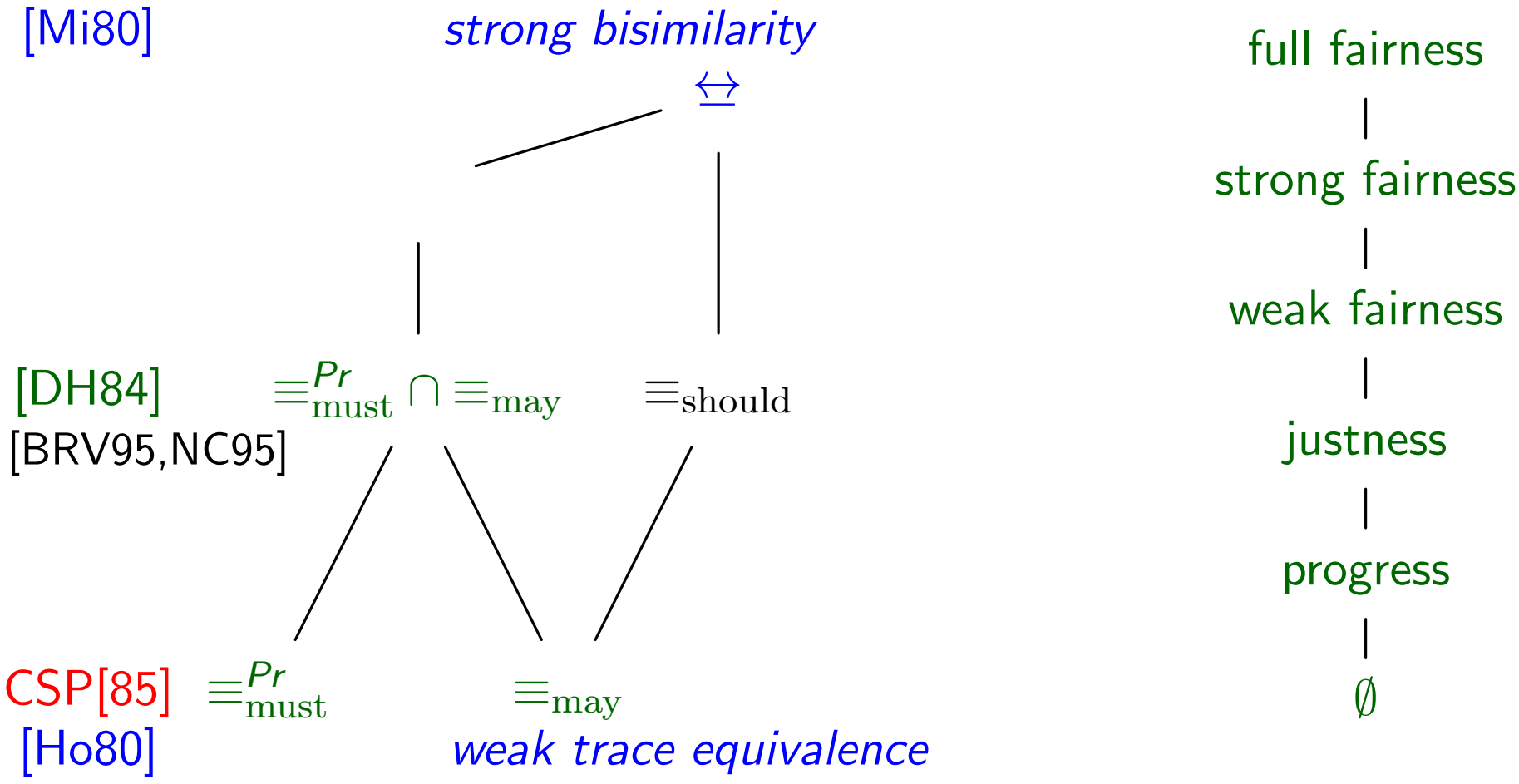
A spectrum of testing preorders and bisimilarities



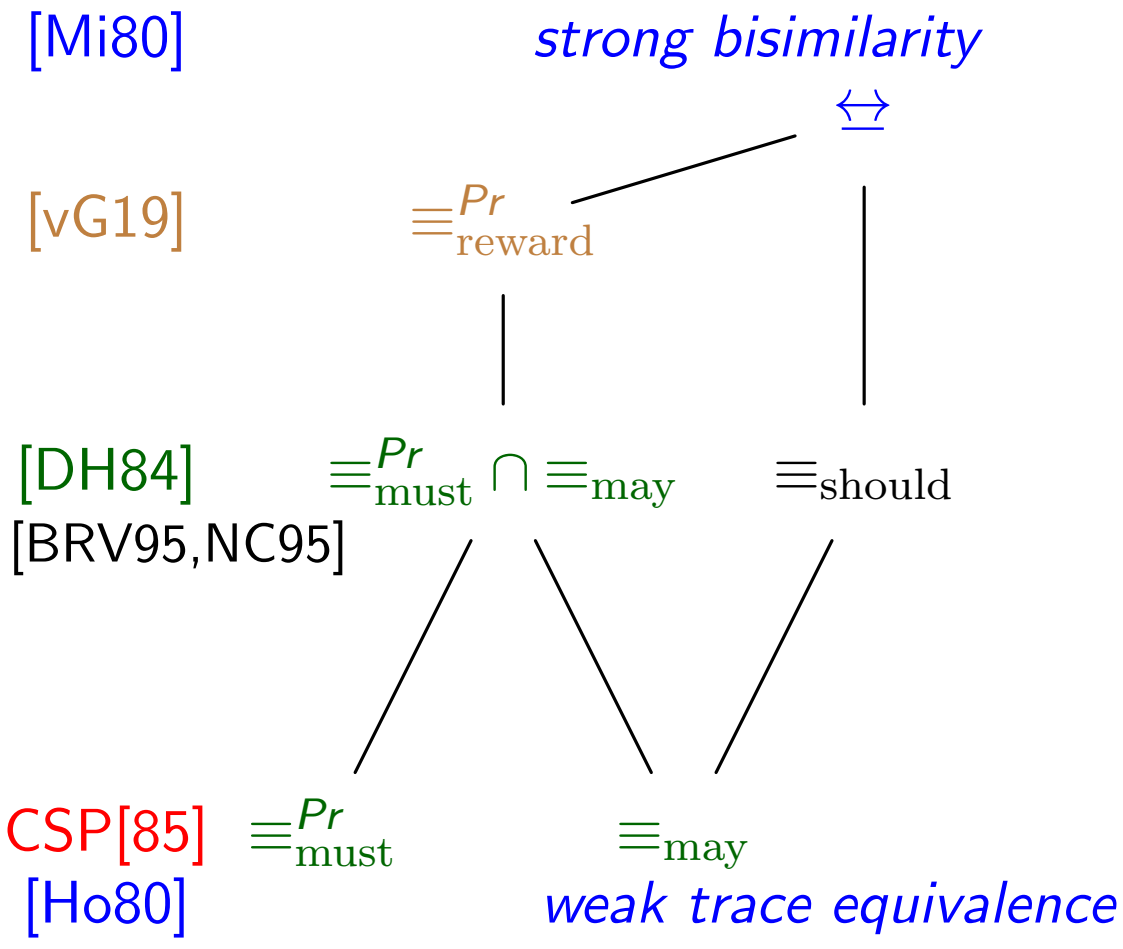
A spectrum of testing preorders and bisimilarities



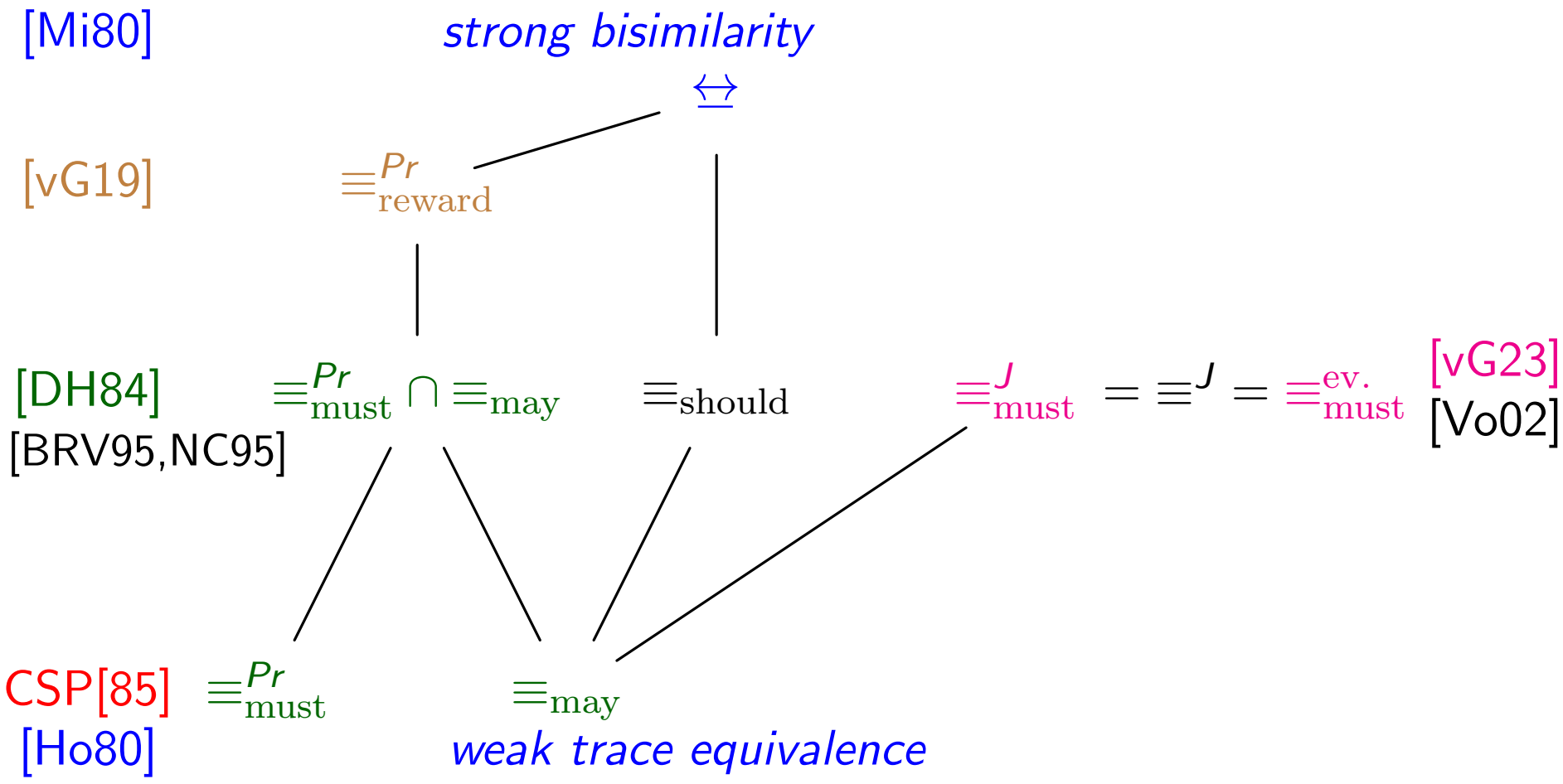
A spectrum of testing preorders and bisimilarities



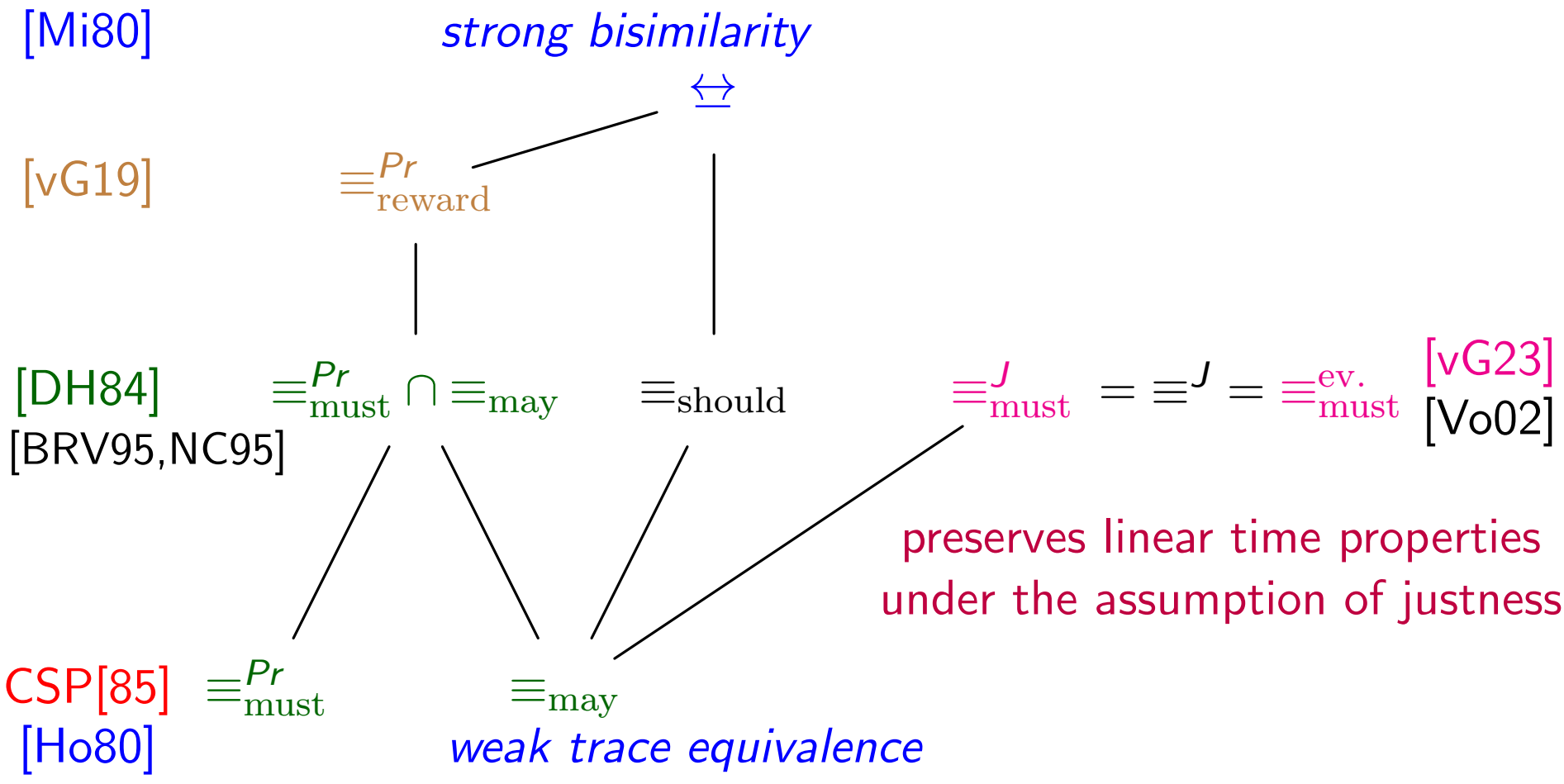
A spectrum of testing preorders and bisimilarities



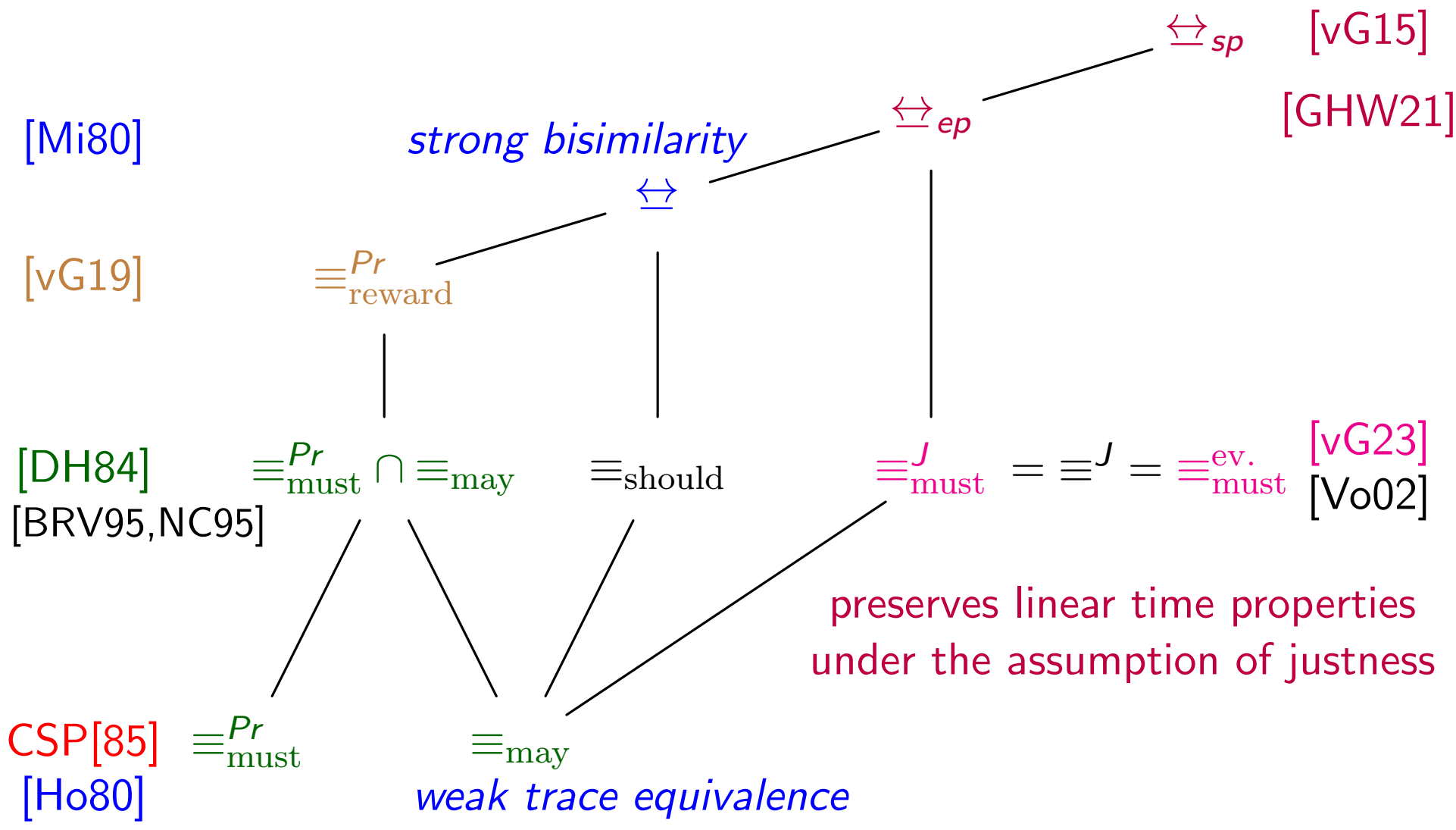
A spectrum of testing preorders and bisimilarities



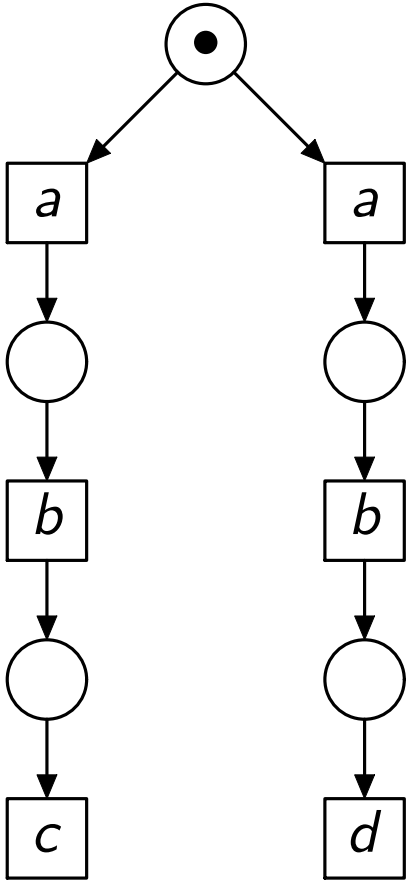
A spectrum of testing preorders and bisimilarities



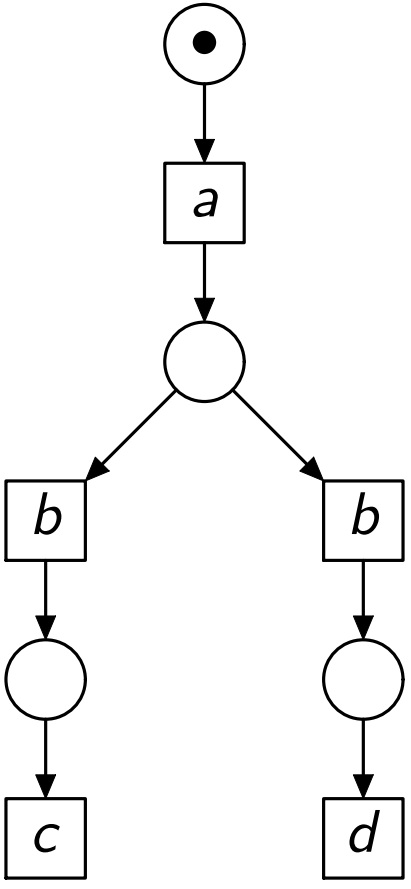
A spectrum of testing preorders and bisimilarities



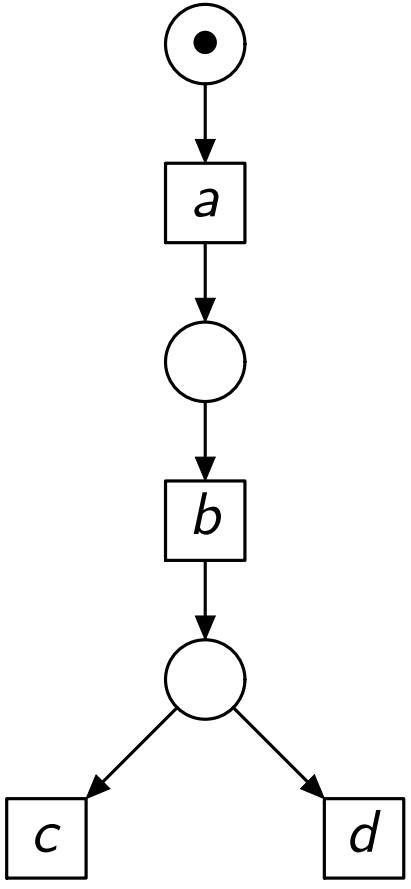
Examples – branching time



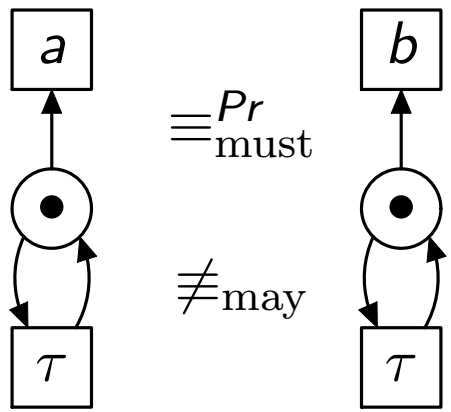
$\equiv_{\text{may}}$
$\equiv^{Pr}_{\text{must}}$
$=^{Pr}_{\text{reward}}$
$\equiv_{\text{should}}$
$\equiv^{J}_{\text{must}}$
$\not\Leftrightarrow$
$\not\Leftrightarrow_{ep}$



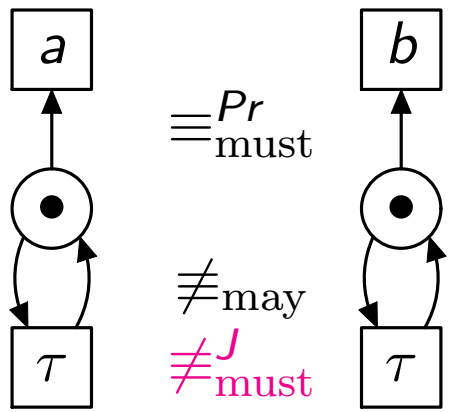
$\equiv_{\text{may}}$
$\not\equiv^{Pr}_{\text{must}}$
$\neq^{Pr}_{\text{reward}}$
$\not\equiv_{\text{should}}$
$\not\equiv^{J}_{\text{must}}$
$\not\Leftrightarrow$
$\not\Leftrightarrow_{ep}$



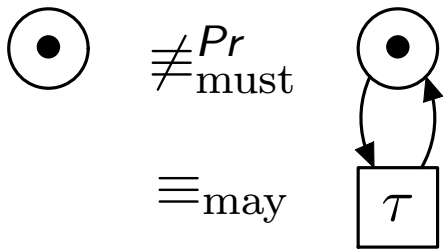
Examples – must testing cannot see past divergence



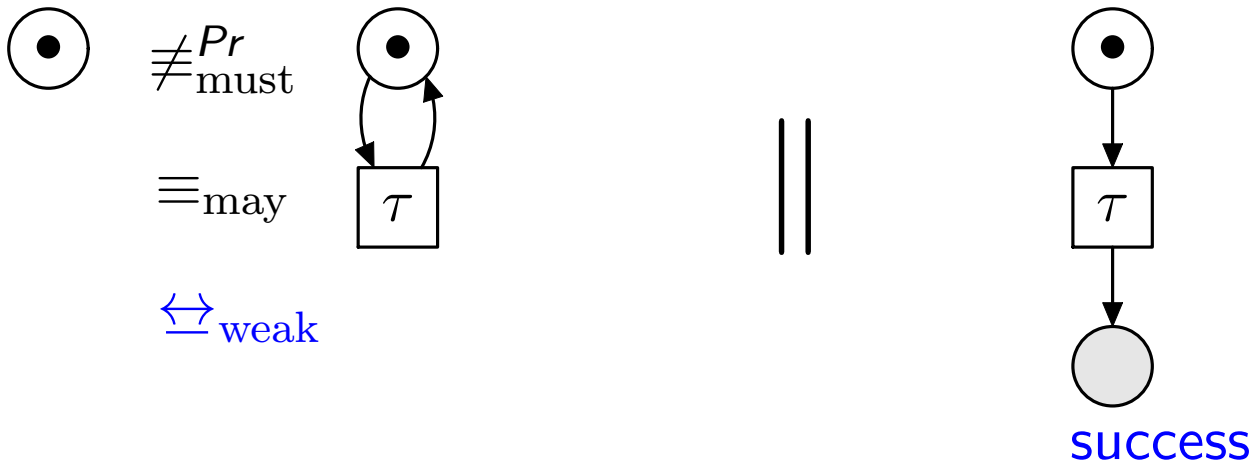
Examples – must testing cannot see past divergence



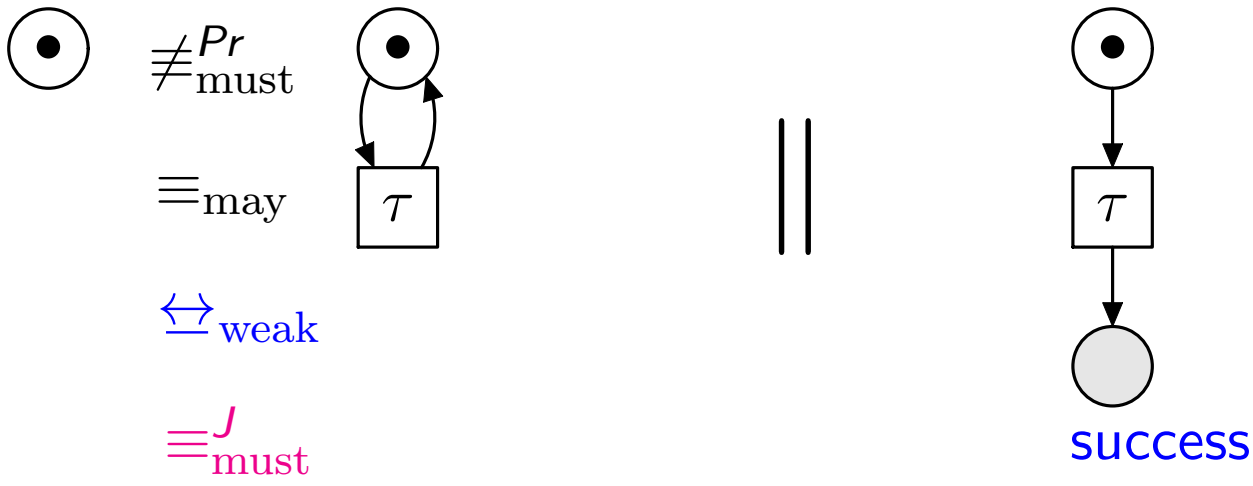
Examples – must testing tells apart deadlock and divergence



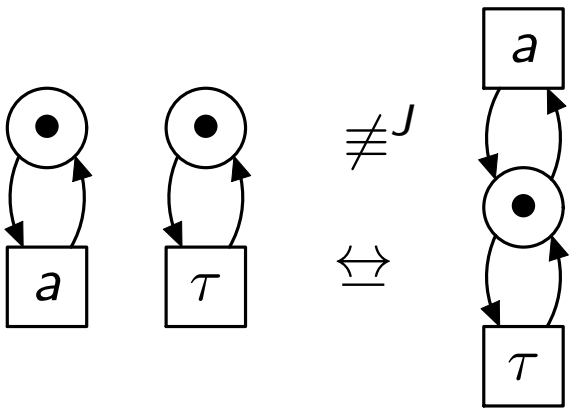
Examples – must testing tells apart deadlock and divergence



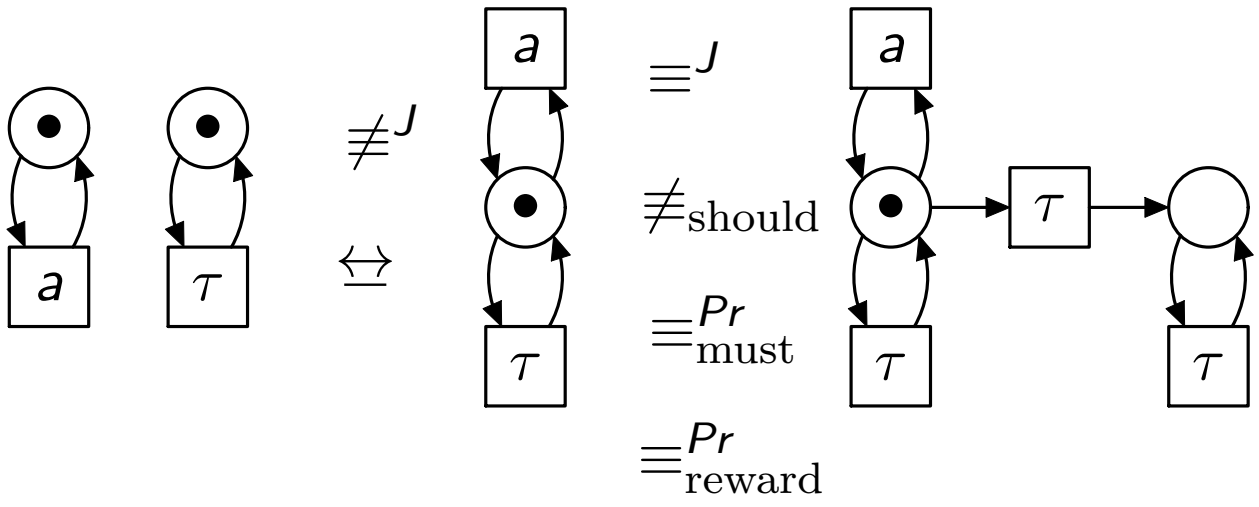
Examples – must testing tells apart deadlock and divergence



Examples – justness and full fairness



Examples – justness and full fairness



full fairness

|

strong fairness

|

weak fairness

|

justness

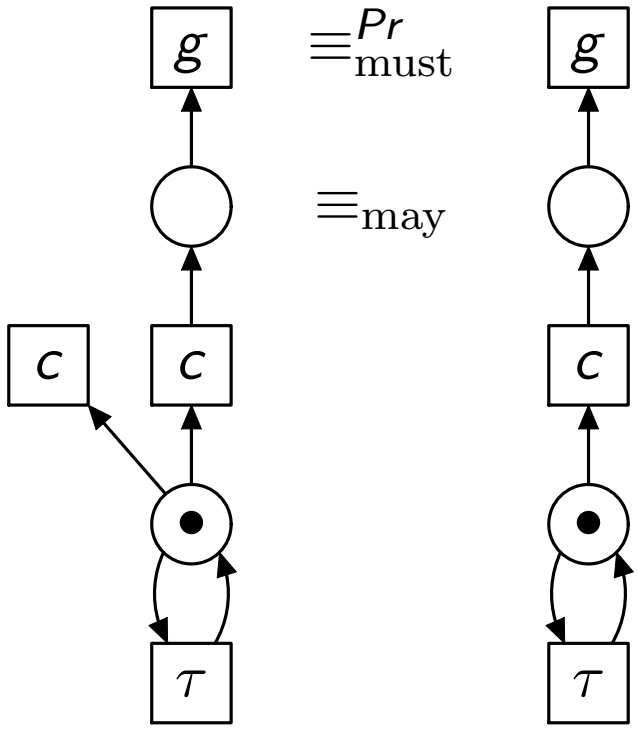
|

progress

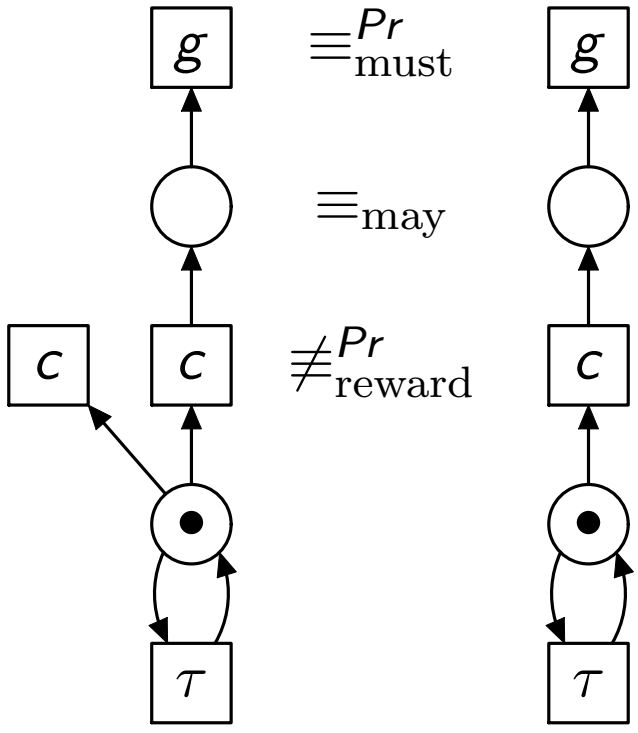
|

∅

Examples – conditional liveness



Examples – conditional liveness



Examples – conditional liveness