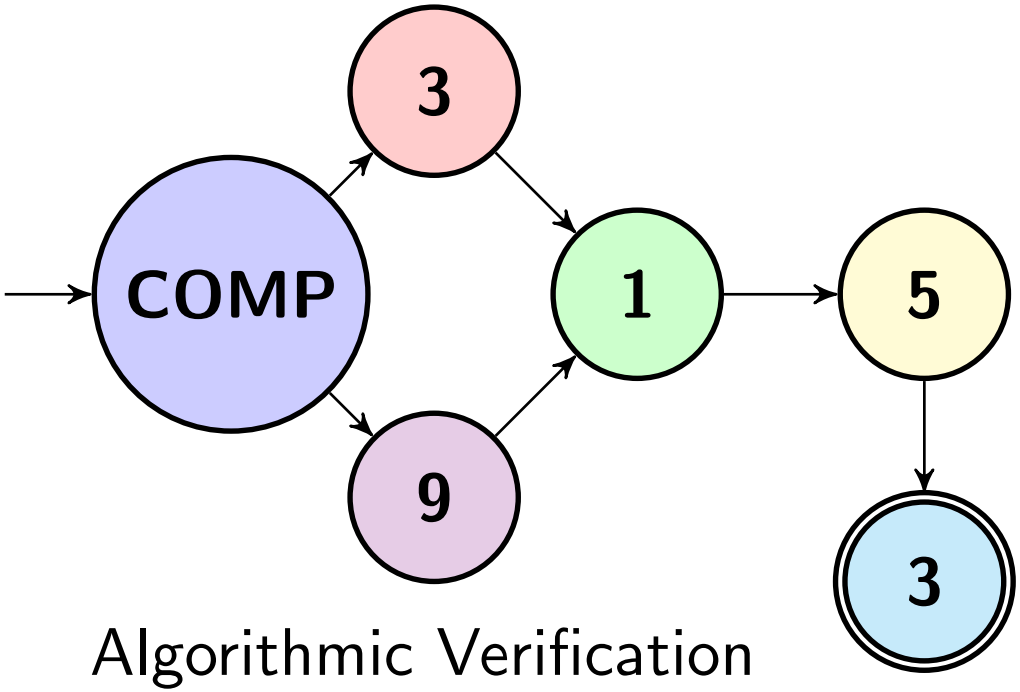


Algorithmic Verification

Simulation and Bisimulation

Dr. Liam O'Connor
CSE, UNSW (for now)
Term 1 2020

Model Equivalence

Let A and B be Kripke structures.

Question

When does $A \models \varphi \Leftrightarrow B \models \varphi$ for all LTL formulae φ?

Model Equivalence

Let A and B be Kripke structures.

Question

When does $A \models \varphi \Leftrightarrow B \models \varphi$ for all LTL formulae φ?
When A and B have the same behaviours. Why?

Liam: prove it on the board

Model Equivalence

Let A and B be Kripke structures.

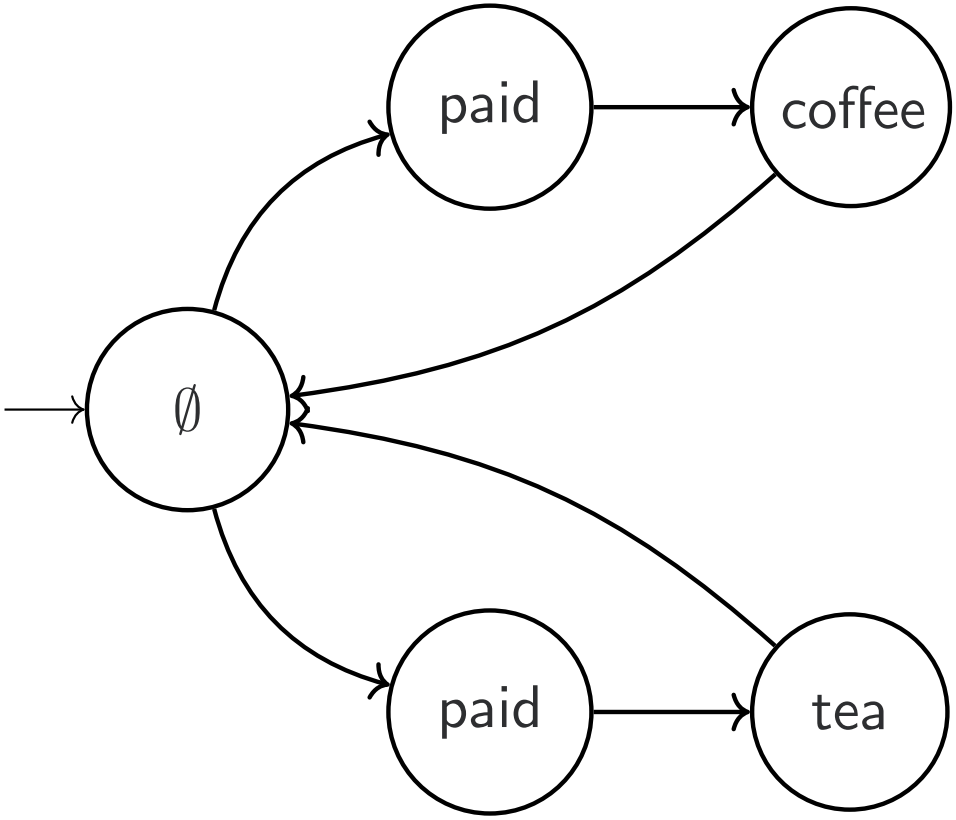
Question

When does $A \models \varphi \Leftrightarrow B \models \varphi$ for all LTL formulae φ?
When A and B have the same behaviours. Why?

Liam: prove it on the board

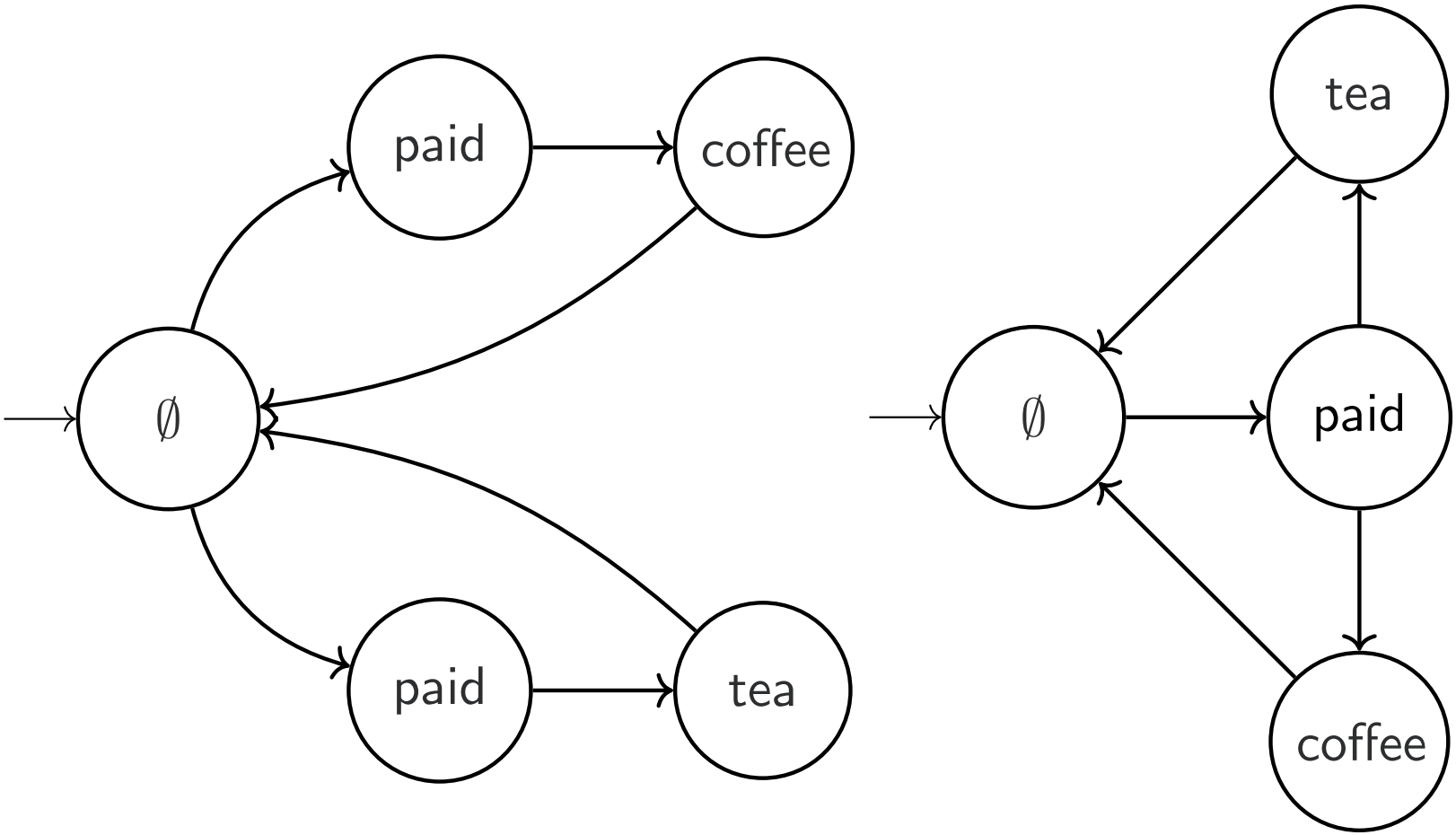
This is called *infinite completed trace equivalence*.

Limitations of Traces



Traces cannot distinguish these two models!

Limitations of Traces



Traces cannot distinguish these two models!

Model Equivalence

Question

When does $A \models \varphi \Leftrightarrow B \models \varphi$ for all CTL formulae φ?

Model Equivalence

Question

When does $A \models \varphi \Leftrightarrow B \models \varphi$ for all CTL formulae φ?
hmm...

Is it (only) when $A = B$ (*graph isomorphism*)?

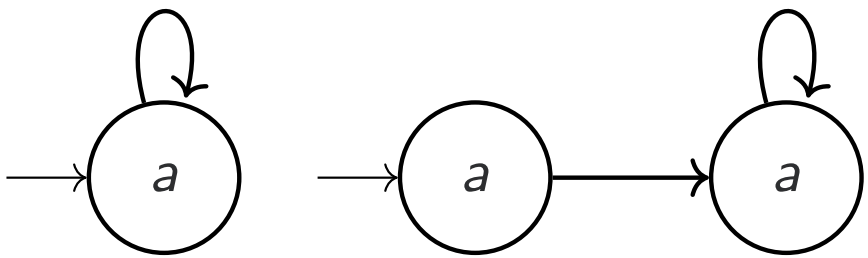
Model Equivalence

Question

When does $A \models \varphi \Leftrightarrow B \models \varphi$ for all CTL formulae φ?

hmm...

Is it (only) when $A = B$ (*graph isomorphism*)?



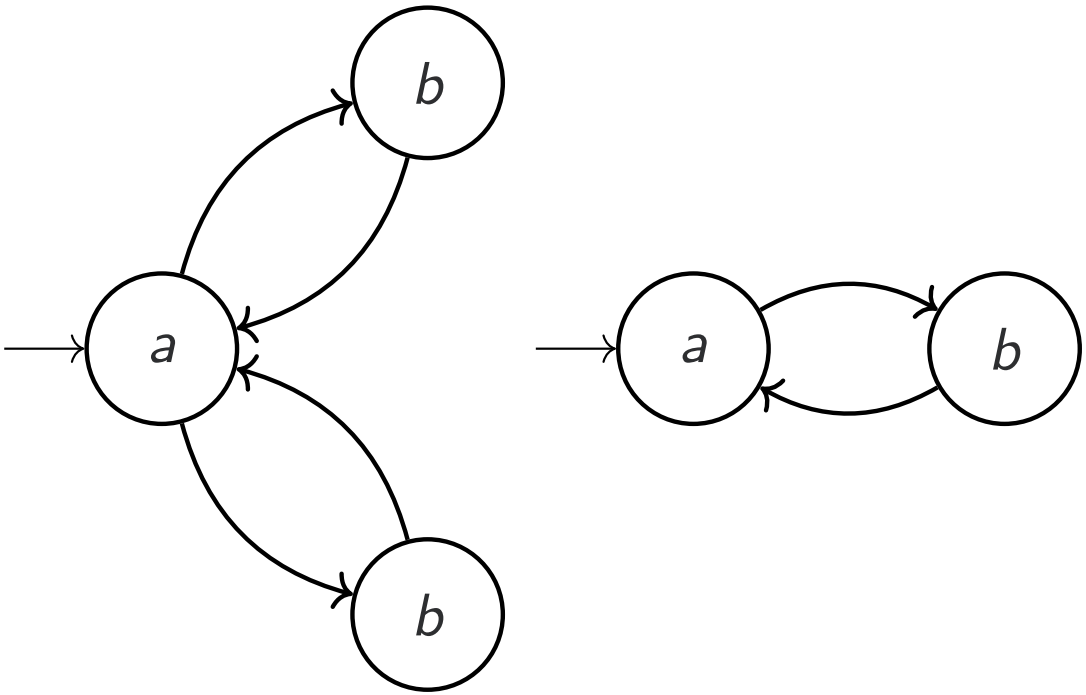
Nope!

Tree Equivalence?

Is it (only) when the two automata have the same computation tree?

Tree Equivalence?

Is it (only) when the two automata have the same computation tree?



Also no!

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$
- If $s \mathcal{R} t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$.

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$
- If $s \mathcal{R} t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$.
- If $s \mathcal{R} t$ and $t \xrightarrow{a} t'$ (with $a \in \Sigma_B, t' \in Q_B$) then there exists a $s' \in Q_A$ such that $s \xrightarrow{a} s'$ and $s' \mathcal{R} t'$.

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$
- If $s \mathcal{R} t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$.
- If $s \mathcal{R} t$ and $t \xrightarrow{a} t'$ (with $a \in \Sigma_B, t' \in Q_B$) then there exists a $s' \in Q_A$ such that $s \xrightarrow{a} s'$ and $s' \mathcal{R} t'$.

Two automata are *bisimulation equivalent* or *bisimilar* iff there exists a bisimulation between their initial states.

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$
- If $s \mathcal{R} t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$.
- If $s \mathcal{R} t$ and $t \xrightarrow{a} t'$ (with $a \in \Sigma_B, t' \in Q_B$) then there exists a $s' \in Q_A$ such that $s \xrightarrow{a} s'$ and $s' \mathcal{R} t'$.

Two automata are *bisimulation equivalent* or *bisimilar* iff there exists a bisimulation between their initial states.

Let's find bisimulations for the previous examples.

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$
- If $s \mathcal{R} t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$.
- If $s \mathcal{R} t$ and $t \xrightarrow{a} t'$ (with $a \in \Sigma_B, t' \in Q_B$) then there exists a $s' \in Q_A$ such that $s \xrightarrow{a} s'$ and $s' \mathcal{R} t'$.

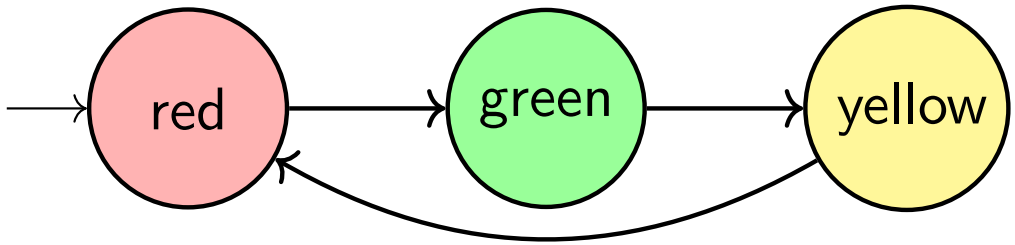
Two automata are *bisimulation equivalent* or *bisimilar* iff there exists a bisimulation between their initial states.

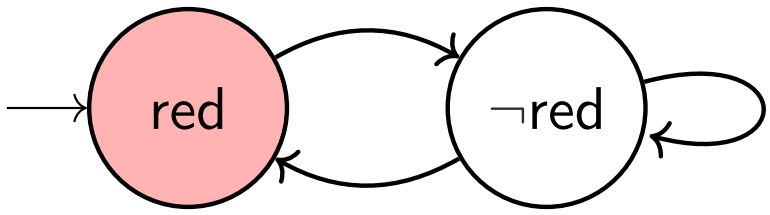
Let's find bisimulations for the previous examples.

Result

For two finitely-branching automata A and B, $A \models \varphi \Leftrightarrow B \models \varphi$ for all CTL formulae φ iff they are *bisimilar*.

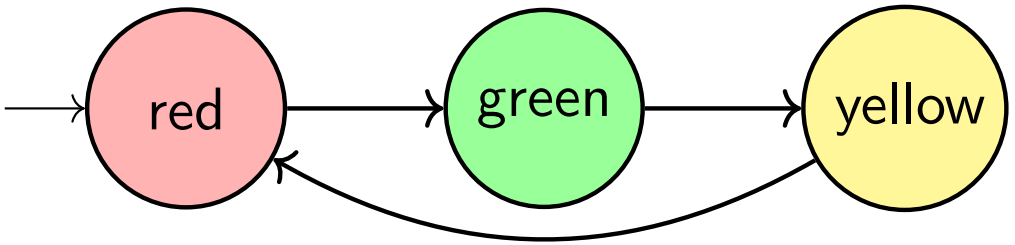
Simulation

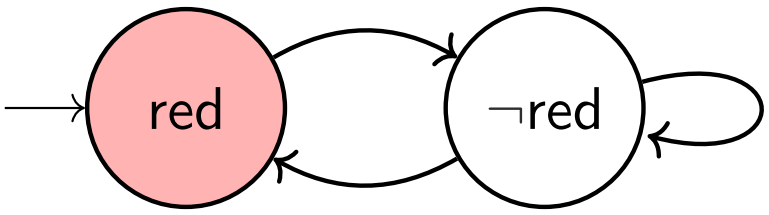




Are these bisimilar?

Simulation





Are these bisimilar?

No, but one simulates the other.

Simulation Relations

Definition

A *simulation* of an automaton C by an automaton A is defined as a relation $\mathcal{S} \subseteq Q_C \times Q_A$ which satisfies:

- If $s \; \mathcal{S} \; t$ then $L_C(s) \cap L_A = L_A(t)$

Simulation Relations

Definition

A *simulation* of an automaton C by an automaton A is defined as a relation $\mathcal{S} \subseteq Q_C \times Q_A$ which satisfies:

- If $s \; \mathcal{S} \; t$ then $L_C(s) \cap L_A = L_A(t)$
- If $s \; \mathcal{S} \; t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_C, s' \in Q_C$) then there exists a $t' \in Q_A$ such that $t \xrightarrow{a} t'$ and $s' \; \mathcal{R} \; t'$.

Simulation Relations

Definition

A *simulation* of an automaton C by an automaton A is defined as a relation $\mathcal{S} \subseteq Q_C \times Q_A$ which satisfies:

- If $s \; \mathcal{S} \; t$ then $L_C(s) \cap L_A = L_A(t)$
- If $s \; \mathcal{S} \; t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_C, s' \in Q_C$) then there exists a $t' \in Q_A$ such that $t \xrightarrow{a} t'$ and $s' \; \mathcal{R} \; t'$.

The automaton A is an *abstraction* of the concrete automaton C iff a A simulates C. This is sometimes written $A \sqsubseteq C$.

Simulation Relations

Definition

A *simulation* of an automaton C by an automaton A is defined as a relation $\mathcal{S} \subseteq Q_C \times Q_A$ which satisfies:

- If $s \; \mathcal{S} \; t$ then $L_C(s) \cap L_A = L_A(t)$
- If $s \; \mathcal{S} \; t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_C, s' \in Q_C$) then there exists a $t' \in Q_A$ such that $t \xrightarrow{a} t'$ and $s' \; \mathcal{R} \; t'$.

The automaton A is an *abstraction* of the concrete automaton C iff a A simulates C. This is sometimes written $A \sqsubseteq C$.

Abstraction and Traces

If $A \sqsubseteq C$, then every trace of C restricted to L_A is a trace of A.

$$\sigma_1\sigma_2\sigma_3 \cdots \in \text{Traces}(C)$$

$$\Rightarrow$$

$$(\sigma_1 \cap L_A)(\sigma_2 \cap L_A)(\sigma_3 \cap L_A) \cdots \in \text{Traces}(A)$$

Essential Property of Simulations

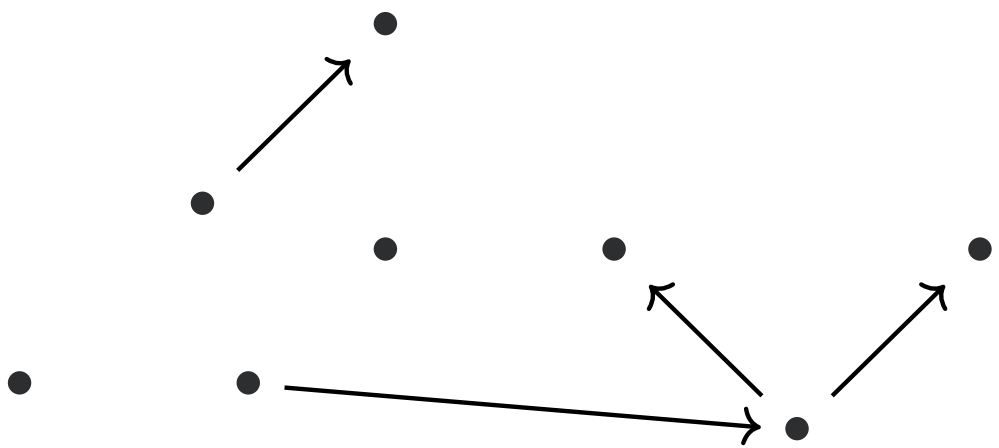
Let $\mathcal{A}$ be a simulation relation, showing that $X \sqsubseteq Y$.
Then for every run $\rho_1\rho_2\rho_3\cdots \in Y$ is a run of X by applying the simulation relation as an *abstraction mapping*:

$$\mathcal{A}(\rho_1)\mathcal{A}(\rho_2)\mathcal{A}(\rho_3)\cdots \in X$$

Essential Property of Simulations

Let $\mathcal{A}$ be a simulation relation, showing that $X \sqsubseteq Y$.
Then for every run $\rho_1\rho_2\rho_3\cdots \in Y$ is a run of X by applying the simulation relation as an *abstraction mapping*:

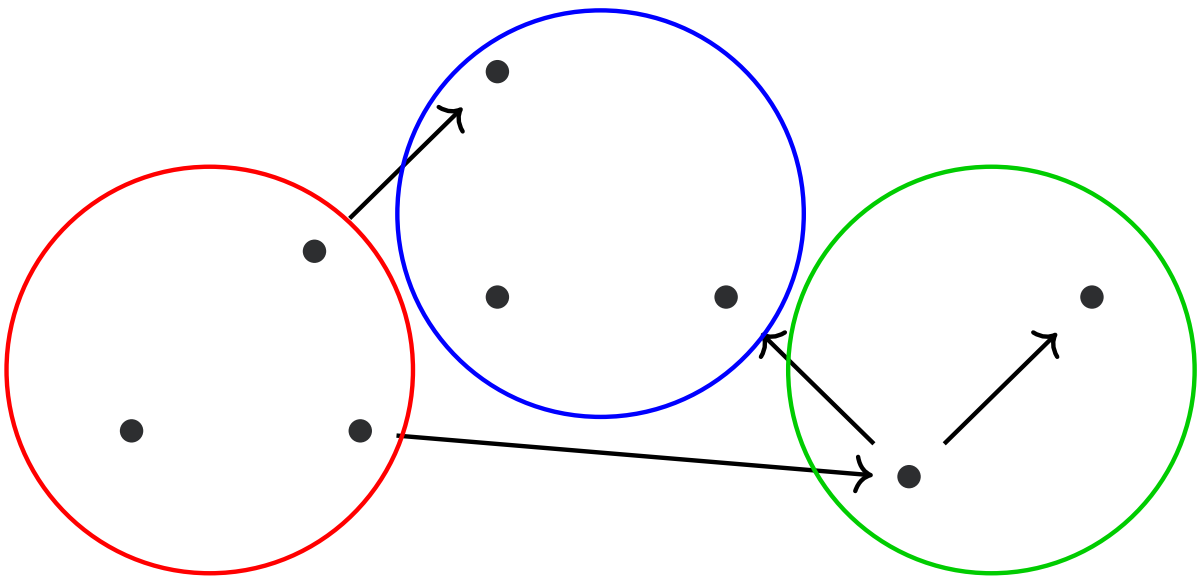
$$\mathcal{A}(\rho_1)\mathcal{A}(\rho_2)\mathcal{A}(\rho_3)\cdots \in X$$



Essential Property of Simulations

Let $\mathcal{A}$ be a simulation relation, showing that $X \sqsubseteq Y$.
Then for every run $\rho_1\rho_2\rho_3\cdots \in Y$ is a run of X by applying the simulation relation as an *abstraction mapping*:

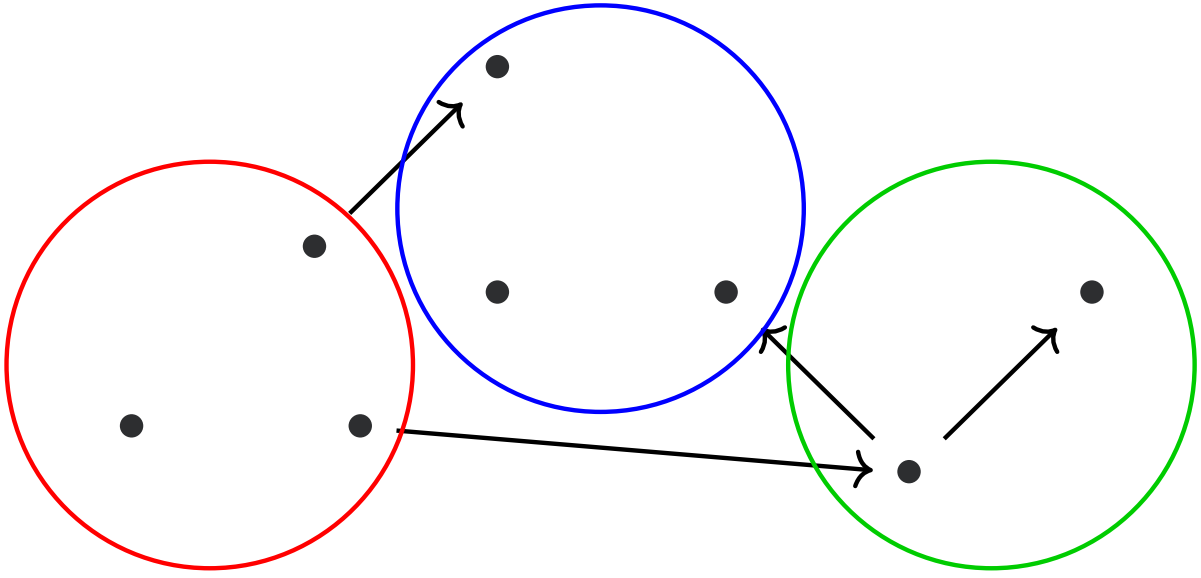
$$\mathcal{A}(\rho_1)\mathcal{A}(\rho_2)\mathcal{A}(\rho_3)\cdots \in X$$

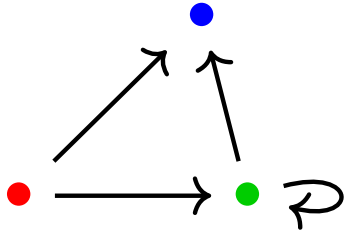


Essential Property of Simulations

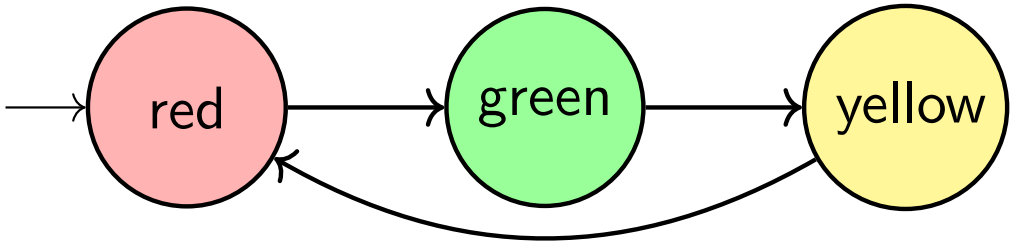
Let $\mathcal{A}$ be a simulation relation, showing that $X \sqsubseteq Y$.
Then for every run $\rho_1\rho_2\rho_3\cdots \in Y$ is a run of X by applying the simulation relation as an *abstraction mapping*:

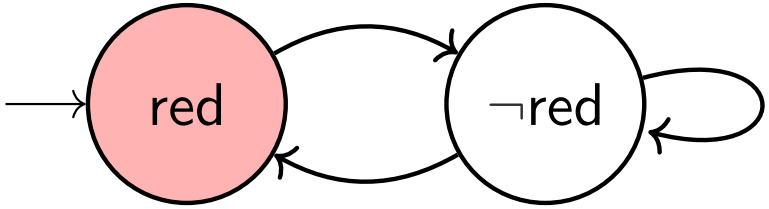
$$\mathcal{A}(\rho_1)\mathcal{A}(\rho_2)\mathcal{A}(\rho_3)\cdots \in X$$

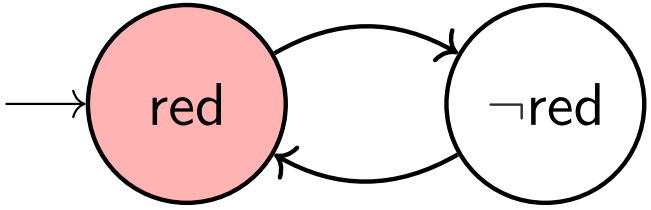


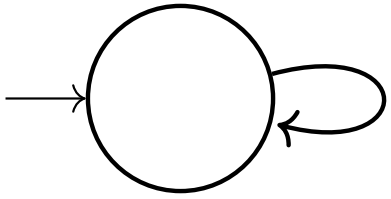


Comparing Automata









What are the simulations between these?

Reducing State Space

We want abstraction to shrink the state space for model checking. To do this, we need a guarantee that any property we prove about an abstraction applies just as well to the concrete model.

Reducing State Space

We want abstraction to shrink the state space for model checking. To do this, we need a guarantee that any property we prove about an abstraction applies just as well to the concrete model.

Universal Properties

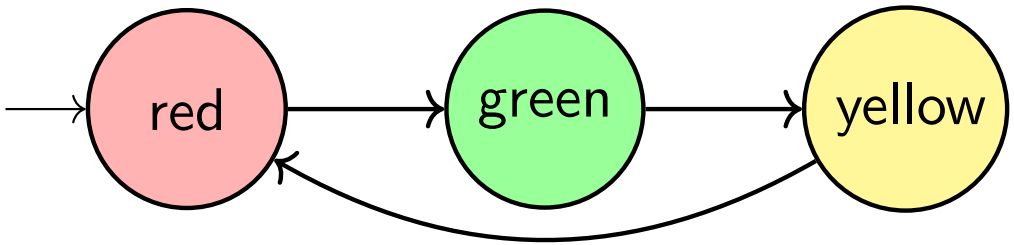
Given $A \sqsubseteq C$, which φ satisfy $A \models \varphi \Rightarrow C \models \varphi$?

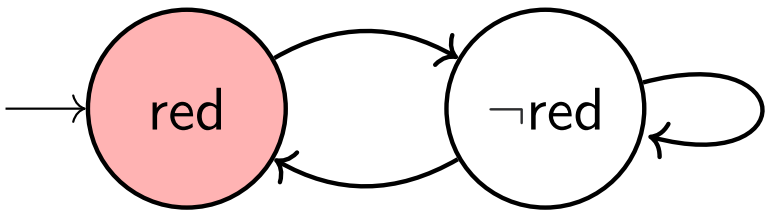
Reducing State Space

We want abstraction to shrink the state space for model checking. To do this, we need a guarantee that any property we prove about an abstraction applies just as well to the concrete model.

Universal Properties

Given $A \sqsubseteq C$, which φ satisfy $A \models \varphi \Rightarrow C \models \varphi$?





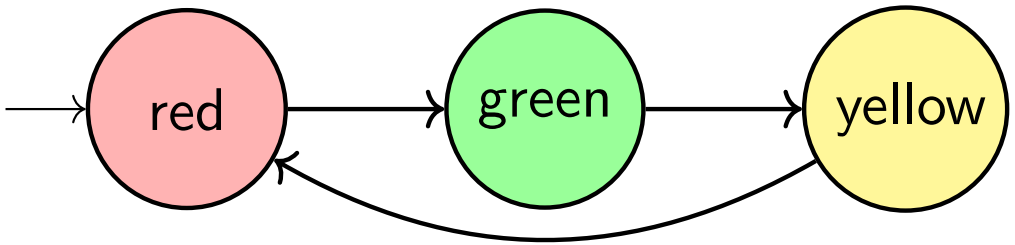
AG AF ¬red?

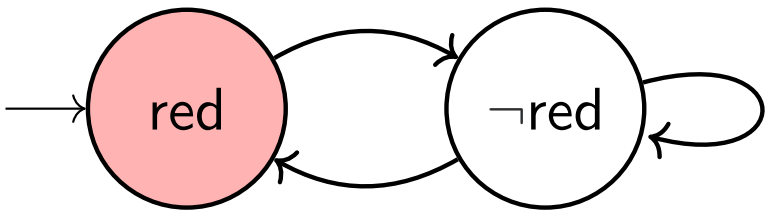
Reducing State Space

We want abstraction to shrink the state space for model checking. To do this, we need a guarantee that any property we prove about an abstraction applies just as well to the concrete model.

Universal Properties

Given $A \sqsubseteq C$, which φ satisfy $A \models \varphi \Rightarrow C \models \varphi$?





AG AF ¬red? Works!

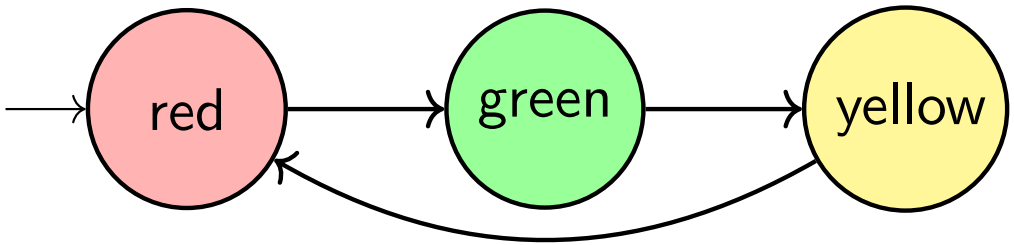
¬**AG AF** red?

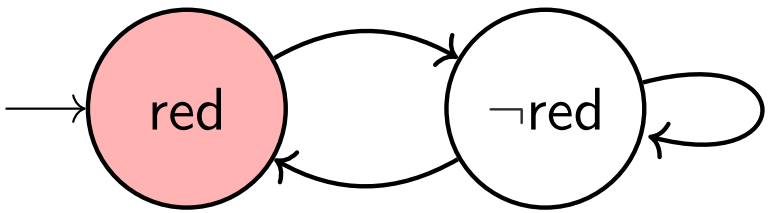
Reducing State Space

We want abstraction to shrink the state space for model checking. To do this, we need a guarantee that any property we prove about an abstraction applies just as well to the concrete model.

Universal Properties

Given $A \sqsubseteq C$, which φ satisfy $A \models \varphi \Rightarrow C \models \varphi$?





AG AF ¬red? Works!

¬**AG AF** red? Doesn't work!

Universal CTL

Negation Normal Form

φ is in negation normal form (NNF), written $\hat{\varphi}$, if all negations are applied only to atomic props. All formulae have a NNF equivalent.

Universal CTL

Negation Normal Form

φ is in negation normal form (NNF), written $\hat{\varphi}$, if all negations are applied only to atomic props. All formulae have a NNF equivalent.

ACTL

φ is a formula in ACTL, the *Universal CTL*, iff its *negation normal form*, $\hat{\varphi}$, does not contain **E**.

Universal CTL

Negation Normal Form

φ is in negation normal form (NNF), written $\hat{\varphi}$, if all negations are applied only to atomic props. All formulae have a NNF equivalent.

ACTL

φ is a formula in ACTL, the *Universal CTL*, iff its *negation normal form*, $\hat{\varphi}$, does not contain **E**.

Example

- **AG**p

Universal CTL

Negation Normal Form

φ is in negation normal form (NNF), written $\hat{\varphi}$, if all negations are applied only to atomic props. All formulae have a NNF equivalent.

ACTL

φ is a formula in ACTL, the *Universal CTL*, iff its *negation normal form*, $\hat{\varphi}$, does not contain **E**.

Example

- **AG**p
- **AG AF**p

Universal CTL

Negation Normal Form

φ is in negation normal form (NNF), written $\hat{\varphi}$, if all negations are applied only to atomic props. All formulae have a NNF equivalent.

ACTL

φ is a formula in ACTL, the *Universal CTL*, iff its *negation normal form*, $\hat{\varphi}$, does not contain **E**.

Example

- **AG**p
- **AG AF**p
- **EF**p

Universal CTL

Negation Normal Form

φ is in negation normal form (NNF), written $\hat{\varphi}$, if all negations are applied only to atomic props. All formulae have a NNF equivalent.

ACTL

φ is a formula in ACTL, the *Universal CTL*, iff its *negation normal form*, $\hat{\varphi}$, does not contain **E**.

Example

- **AG**p
- **AG AF**p
- **EF**p — Nope!

Negation Normal Form

$$
\begin{aligned}
\neg \mathbf{AF}\varphi &\equiv \mathbf{EG}\neg\varphi \\
\neg \mathbf{EF}\varphi &\equiv \mathbf{AG}\neg\varphi \\
\neg \mathbf{AG}\varphi &\equiv \mathbf{EF}\neg\varphi \\
\neg \mathbf{EG}\varphi &\equiv \mathbf{AF}\neg\varphi \\
\neg \mathbf{AX}\varphi &\equiv \mathbf{EX}\neg\varphi \\
\neg \mathbf{EX}\varphi &\equiv \mathbf{AX}\neg\varphi \\
\neg \mathbf{E}(\varphi\ \mathbf{U}\ \psi) &\equiv
\end{aligned}
$$

Negation Normal Form

$$
\begin{aligned}
\neg \mathbf{AF}\varphi &\equiv \mathbf{EG}\neg\varphi \\
\neg \mathbf{EF}\varphi &\equiv \mathbf{AG}\neg\varphi \\
\neg \mathbf{AG}\varphi &\equiv \mathbf{EF}\neg\varphi \\
\neg \mathbf{EG}\varphi &\equiv \mathbf{AF}\neg\varphi \\
\neg \mathbf{AX}\varphi &\equiv \mathbf{EX}\neg\varphi \\
\neg \mathbf{EX}\varphi &\equiv \mathbf{AX}\neg\varphi \\
\neg \mathbf{E}(\varphi\ \mathbf{U}\ \psi) &\equiv \mathbf{A}(\neg\varphi\ \mathbf{R}\ \neg\psi) \\
\neg \mathbf{A}(\varphi\ \mathbf{U}\ \psi) &\equiv \mathbf{E}(\neg\varphi\ \mathbf{R}\ \neg\psi)
\end{aligned}
$$

Negation Normal Form

$$
\begin{aligned}
\neg \mathbf{AF}\varphi &\equiv \mathbf{EG}\neg\varphi \\
\neg \mathbf{EF}\varphi &\equiv \mathbf{AG}\neg\varphi \\
\neg \mathbf{AG}\varphi &\equiv \mathbf{EF}\neg\varphi \\
\neg \mathbf{EG}\varphi &\equiv \mathbf{AF}\neg\varphi \\
\neg \mathbf{AX}\varphi &\equiv \mathbf{EX}\neg\varphi \\
\neg \mathbf{EX}\varphi &\equiv \mathbf{AX}\neg\varphi \\
\neg \mathbf{E}(\varphi\ \mathbf{U}\ \psi) &\equiv \mathbf{A}(\neg\varphi\ \mathbf{R}\ \neg\psi) \\
\neg \mathbf{A}(\varphi\ \mathbf{U}\ \psi) &\equiv \mathbf{E}(\neg\varphi\ \mathbf{R}\ \neg\psi)
\end{aligned}
$$

Release Operator

The temporal operator $\varphi\ \mathbf{R}\ \psi$ says that ψ will not become false unless φ happens first.

$$\sigma \models \varphi\ \mathbf{R}\ \psi \quad \Leftrightarrow \quad \forall n \geq 0.\ (\forall 0 \leq k < n.\ \sigma|_k \not\models \varphi) \Rightarrow \sigma|_n \models \psi$$

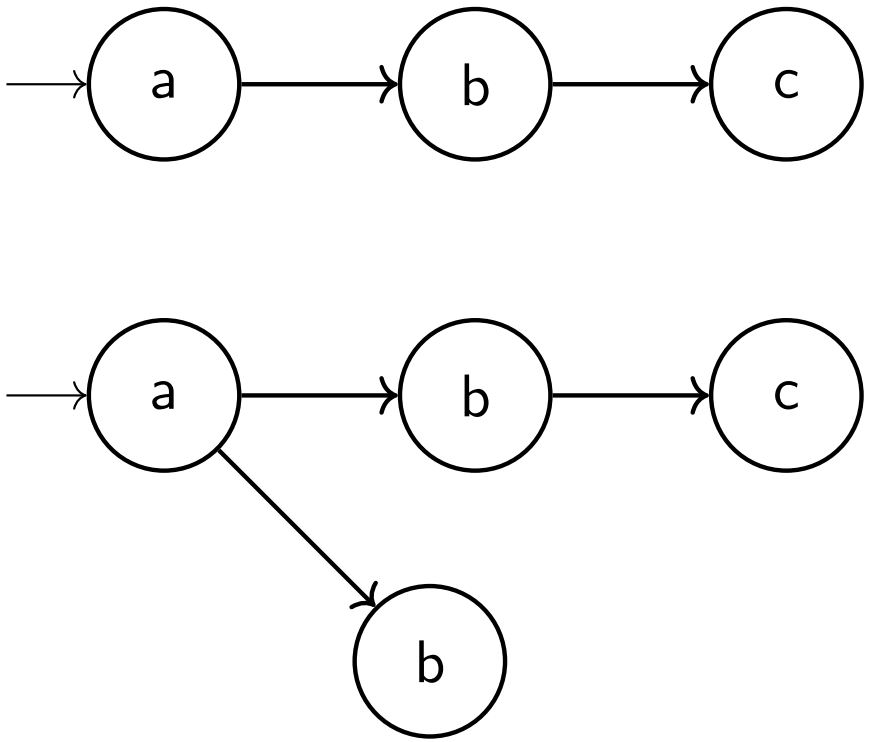
A and **E** variants in CTL follow the usual pattern.

Bisimulation and simulation

Suppose that $A \sqsubseteq B$ and $B \sqsubseteq A$. Does that mean A is bisimilar to B?

Bisimulation and simulation

Suppose that $A \sqsubseteq B$ and $B \sqsubseteq A$. Does that mean A is bisimilar to B?



Nope! This is another equivalence called *simulation equivalence*. Because of the abstraction result, ACTL is the logic that characterises simulation equivalence.

The Linear-time Branching-time Spectrum

Coarseness of Equivalences

- Graph isomorphism is finer (distinguishes more models) in than bisimilarity.

The Linear-time Branching-time Spectrum

Coarseness of Equivalences

- Graph isomorphism is finer (distinguishes more models) in than bisimilarity.
- Bisimilarity is finer than simulation equivalence.

The Linear-time Branching-time Spectrum

Coarseness of Equivalences

- Graph isomorphism is finer (distinguishes more models) in than bisimilarity.
- Bisimilarity is finer than simulation equivalence.
- Bisimilarity is finer that completed infinite trace equivalence.

The Linear-time Branching-time Spectrum

Coarseness of Equivalences

- Graph isomorphism is finer (distinguishes more models) in than bisimilarity.
- Bisimilarity is finer than simulation equivalence.
- Bisimilarity is finer that completed infinite trace equivalence.
- Partial trace equivalence (sets of finite-length traces) is coarser than all of the above.

The Linear-time Branching-time Spectrum

Coarseness of Equivalences

- Graph isomorphism is finer (distinguishes more models) in than bisimilarity.
- Bisimilarity is finer than simulation equivalence.
- Bisimilarity is finer that completed infinite trace equivalence.
- Partial trace equivalence (sets of finite-length traces) is coarser than all of the above.

There are many, many more equivalences.

The Linear-time Branching-time Spectrum

Coarseness of Equivalences

- Graph isomorphism is finer (distinguishes more models) in than bisimilarity.
- Bisimilarity is finer than simulation equivalence.
- Bisimilarity is finer that completed infinite trace equivalence.
- Partial trace equivalence (sets of finite-length traces) is coarser than all of the above.

There are many, many more equivalences.

Rob van Glabbeek categorised all of these equivalences and more into the *linear-time branching-time spectrum*, which is a major focus of his course at this university, COMP6752.

Bibliography

- Baier/Katoen, Sections 7.1 (parts), 7.2 (parts), 7.4, 7.5, 7,6, 7.7
- Rob van Glabbeek, The Linear-Time Branching-Time Spectrum I, Handbook of Process Algebra p. 3-99, Elsevier.
- Rob van Glabbeek, COMP6752 course notes.