

Satisfiability Checking

Bounded Model Checking

Prof. Dr. Erika Ábrahám

Theory of Hybrid Systems
Informatik 2

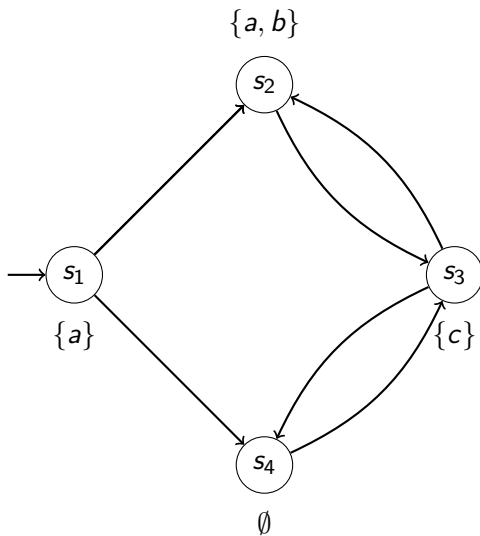
WS 10/11

1 Transition Systems

2 Logic

3 Bounded Model Checking

Kripke Structure



Definition

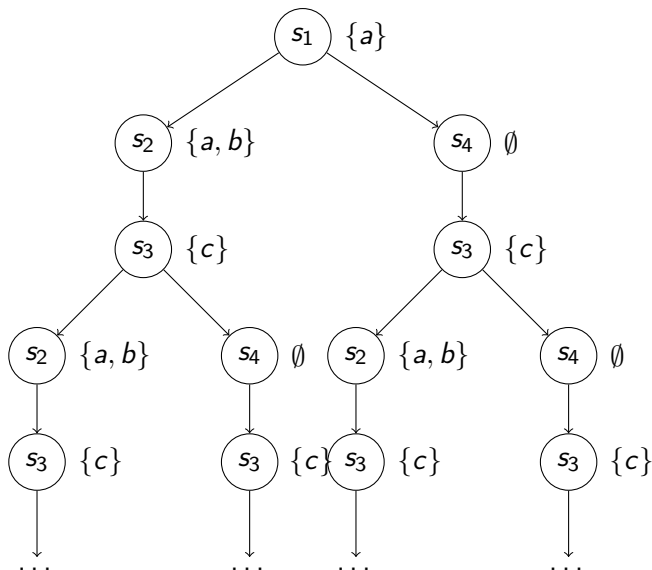
Let AP be a finite set of atomic propositions. A Kripke structure is a tuple $M = (S, s_{\text{init}}, T, L)$ with

- S a finite set of states,
- $s_{\text{init}} \in S$ an initial state,
- $T \subseteq S \times S$ a transition relation,
- $L : S \rightarrow 2^{AP}$ a labeling function
(2^{AP} denotes the powerset over AP).

The labeling function attaches information to the system: for a state $s \in S$ the set $L(s)$ consists of those atomic propositions that hold in s .

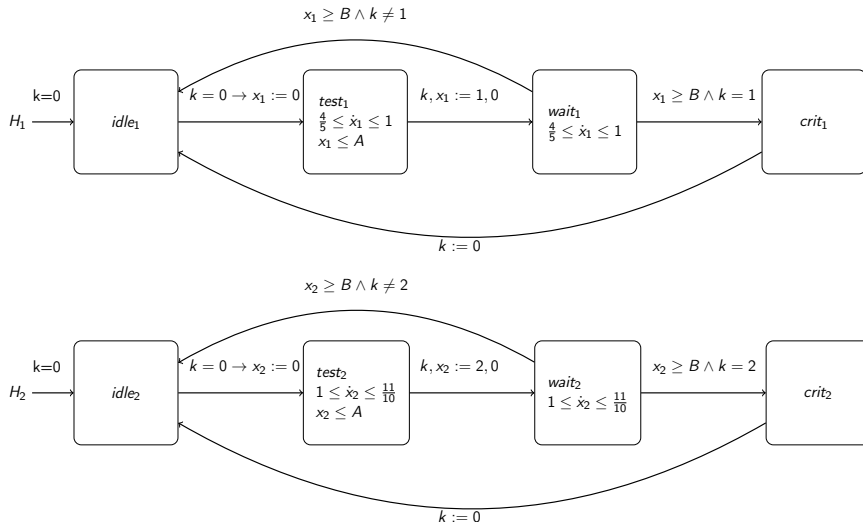
- An (infinite) path $\pi = s_0 s_1 s_2 \dots$ of a Kripke structure $M = (S, s_{\text{init}}, T, L)$ is a sequence of states such that
 - $s_0 = s_{\text{init}}$ and
 - for all $i \geq 0$, $(s_i, s_{i+1}) \in T$.
- We write $\pi(j)$ for the j th state (starting with 0) of the path π .
- π_j denotes the postfix of π starting at $\pi(j)$.
- A finite path of M is a finite prefix of an infinite path of M .
- For a finite path $\pi = s_0 \dots s_k$ we define $|\pi| = k$.
- The behaviour of M is given by the set of all of its infinite paths.

Kripke Structure: Semantics



Fischer's mutual exclusion protocol

There are also more complex systems we want to deal with later.



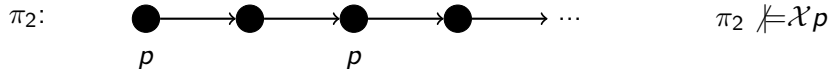
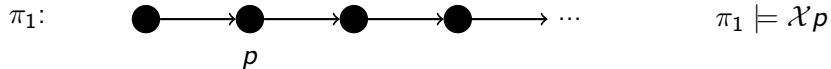
Syntax of the Linear-Time Temporal Logic (LTL):

$$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg \varphi \mid \mathcal{X}\varphi \mid \varphi \mathcal{U} \varphi \mid \mathcal{F}\varphi \mid \mathcal{G}\varphi$$

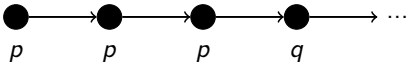
- $a \in AP$: atomic proposition
- \mathcal{X} : next time operator
- \mathcal{U} : until operator

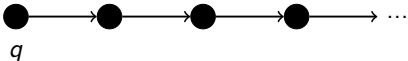
Syntactic sugar:

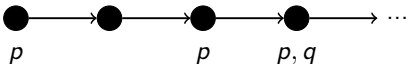
- $\vee, \rightarrow, \iff, \dots$
- \mathcal{F} : finally (eventually) operator ($\mathcal{F}\varphi := true \mathcal{U} \varphi$)
- \mathcal{G} : globally (always) operator ($\mathcal{G}\varphi := \neg(true \mathcal{U} \neg \varphi)$)

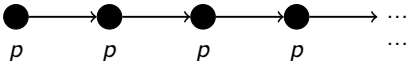


LTl Semantics - Until

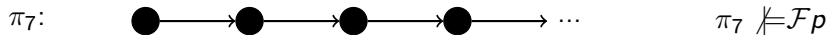
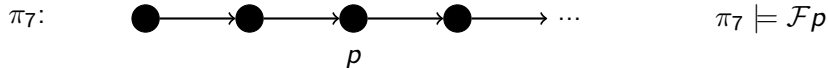
π_3 :  $\pi_3 \models p\mathcal{U}q$

π_4 :  $\pi_4 \models p\mathcal{U}q$

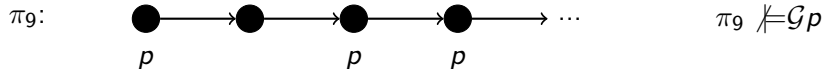
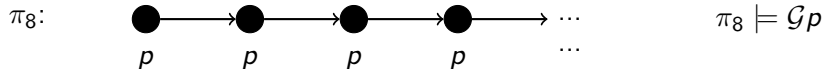
π_5 :  $\pi_5 \not\models p\mathcal{U}q$

π_6 :  $\pi_6 \not\models p\mathcal{U}q$

LTl Semantics - Eventually



LTl Semantics - Always



Definition (LTL Semantics)

$\pi \models p$	iff	$p \in L(\pi(0))$
$\pi \models \varphi_1 \wedge \varphi_2$	iff	$\pi \models \varphi_1$ and $\pi \models \varphi_2$
$\pi \models \neg \varphi$	iff	$\pi \not\models \varphi$
$\pi \models \mathcal{X}\varphi$	iff	$\pi_1 \models \varphi$
$\pi \models \varphi_1 \mathcal{U} \varphi_2$	iff	$\pi_i \models \varphi_2$ for some $i \geq 0$ and $\pi_j \models \varphi_1$ for all $0 \leq j < i$
$\pi \models \mathcal{F}\varphi$	iff	$\pi_i \models \varphi$ for some $i \geq 0$
$\pi \models \mathcal{G}\varphi$	iff	$\pi_i \models \varphi$ for all $i \geq 0$

$$M \models A\varphi$$

If all infinite paths of a Kripke structure M satisfy a property φ , then we say that **the property holds for M** .

$$M \models E\neg\varphi$$

If there is an infinite path of a Kripke structure M that does not satisfy a property φ , then we say that **M violates the property φ** .

$$M \models_k E\neg\varphi$$

Also **finite paths** can violate a property, if they contain enough information to assure the existence of an infinite path violating the property.

Early 1980s: First implementations of Model Checking as verification technique

- Explicit representations of the transition graphs
- **Problem:** Due to the state explosion not applicable for most industrial settings

1990: Symbolic Model Checking

- BDDs represent *characteristic functions* of state sets
- **Problem:** Building the BDD may be expensive

1999: Bounded Model Checking *by Biere et al.*

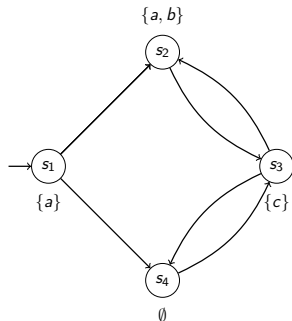
- Incrementally finite paths of certain length are checked by SAT-Solver
- **Problem:** No completeness is guaranteed

- Given a Kripke structure M and an LTL property φ , a **counterexample** is a path of M violating φ .
- If a system is buggy, counterexamples are extremely important for detecting and fixing the error.
- **Bounded model checking (BMC)** is a technique to search for finite counterexamples, not only for Kripke structures, but also for more complex systems.

Finite and infinite Counterexamples

Property: $\mathcal{GF}a$

Negation: $\neg\mathcal{GF}a = \mathcal{FG}\neg a$



Infinite counterexample:

$s_1 \ s_4 \ s_3 \ s_4 \ s_3 \ s_4 \ s_3 \ \dots$

Finite counterexample:

$s_1 \ \boxed{s_4} \ s_3 \ \boxed{s_4}$

“Loop detected”

Definition of \models_k

Satisfaction relation for **finite paths**

$\pi \models_k^i \varphi$: the finite segment of π consisting of its i th to k th states satisfies φ

$\pi \models_k \varphi$: $\pi \models_k^0 \varphi$

Definition of \models_k

Satisfaction relation for **finite paths with a loop**

Definition

For $l \leq k$ we call an infinite path π a **(k, l) -loop** iff $T(\pi(k), \pi(l))$ and $\pi = u \cdot v^\omega$ with $u = \pi(0) \dots \pi(l-1)$ and $v = \pi(l) \dots \pi(k)$.

We call π a **k -loop** iff π is a (k, l) -loop for some $0 \leq l \leq k$.

Definition (Bounded Semantics for a loop)

Let $k \geq 0$ and let π be a k -loop. Then an LTL formula φ is **valid along π with bound k** ($\pi \models_k \varphi$) iff $\pi \models \varphi$.

Definition of \models_k

Definition (Bounded Semantics without a loop)

Let $k \geq 0$ and let π be path that is not a k -loop. Then an LTL formula φ is valid along π with bound k ($\pi \models_k \varphi$) iff $\pi \models_k^0 \varphi$, where

$$\begin{aligned}\pi \models_k^i a & : a \in L(\pi(i)) \\ \pi \models_k^i \neg a & : a \notin L(\pi(i)) \\ \pi \models_k^i \varphi_1 \wedge \varphi_2 & : \pi \models_k^i \varphi_1 \text{ and } \pi \models_k^i \varphi_2 \\ \pi \models_k^i \mathcal{X}\varphi & : i < k \text{ and } \pi \models_k^{i+1} \varphi \\ \pi \models_k^i \varphi_1 \mathcal{U} \varphi_2 & : \pi \models_k^j \varphi_2 \text{ for some } i \leq j \leq k \text{ and} \\ & \quad \pi \models_k^n \varphi_1 \text{ for all } i \leq n < j \\ \pi \models_k^i \mathcal{F}\varphi & : \pi \models_k^j \varphi \text{ for some } i \leq j \leq k \\ \pi \models_k^i \mathcal{G}\varphi & : \text{false}\end{aligned}$$

Properties of \models_k

Lemma

Let φ be an LTL formula and let π be a path. Then

$$\pi \models_k \varphi \Rightarrow \pi \models \varphi.$$

Lemma

Let φ be an LTL formula and let M be a Kripke structure. Then

$$M \models \mathbf{E}\varphi \Rightarrow \exists k \geq 0. M \models_k \mathbf{E}\varphi.$$

Overview:

- Construction of a Boolean formula φ describing a finite path
 - through the underlying system
 - of length k , starting with 0,
 - and reaching a certain state of interest, i.e., violating a property.
- A SAT-solver searches for a satisfying assignment of φ
- If SAT, the resulting assignment describes a counterexample
- If UNSAT, k is incremented and the procedure starts again

Counterexamples of length k for a Kripke structure M and an LTL formula φ can be described by

$$\llbracket M, \varphi \rrbracket_k = I(s_0) \wedge T(s_0, s_1) \wedge \dots \wedge T(s_{k-1}, s_k) \wedge \neg \text{Prop}(s_0, \dots, s_k)$$

$\llbracket M, \varphi \rrbracket_k$ is satisfiable \iff there exists a finite counterexample of length k

\rightarrow check $\llbracket M, \varphi \rrbracket_k$ incrementally for $k = 0, 1, \dots$ using a suitable solver

$$\llbracket M, \varphi \rrbracket_k = I(s_0) \wedge T(s_0, s_1) \wedge \dots \wedge T(s_{k-1}, s_k) \wedge \neg \text{Prop}(s_0 \dots s_k)$$

How to build this formula?

- I and T are (nearly) straightforward for Kripke structures. We build a sub-formula describing initial paths of length k :

$$\llbracket M \rrbracket_k := I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1})$$

- This formula is called the **unfolding of the transition relation**

$$\llbracket M, \varphi \rrbracket_k = I(s_0) \wedge T(s_0, s_1) \wedge \dots \wedge T(s_{k-1}, s_k) \wedge \neg \text{Prop}(s_0 \dots s_k)$$

How to build this formula?

- To get a counterexample for an LTL formula φ we have to find a **witness** for $\neg\varphi$.
- This will be encoded for paths of length k within the formula $\neg\text{Prop}(s_0, \dots, s_k)$
- The translation of the formula depends on the fact whether the considered path has a loop or not.

Encoding of loops

Loop condition: Is there an transition from s_k to a previous state?

Loop successor: Successor state of a state inside a loop

Definition

The loop condition L_k is true iff there exists a back loop from state s_k to a previous state or to itself: $L_k := \bigvee_{l=0}^k T(s_k, s_l)$

Definition

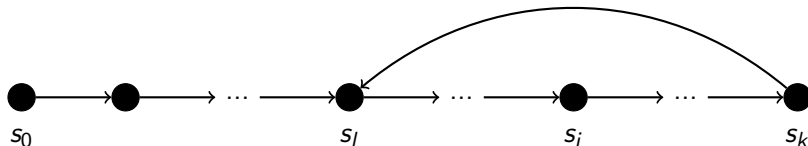
Let k, l and i be non-negative integers with $l, i \leq k$.

- $\text{succ}(i) := i + 1$, if i is inside a (k, l) -loop, i.e. $i < k$
- $\text{succ}(i) := l$ for $i = k$

Encoding of loops - Always

- Given: LTL formula φ and path π with (k, l) -loop
- Recursive translation over the sub-terms of φ and states in π
- Introduce intermediate formula of the form ${}_l\llbracket \cdot \rrbracket_k^i$
 - l start-state of the loop
 - k bound
 - i current position
- Translation rule for $\mathcal{G}\varphi$:

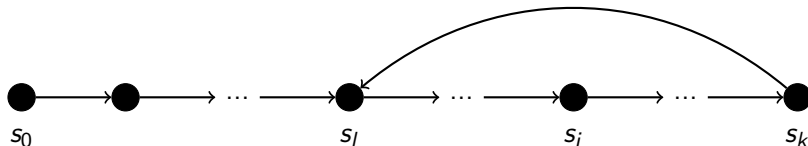
$${}_l\llbracket \mathcal{G}\varphi \rrbracket_k^i := {}_l\llbracket \varphi \rrbracket_k^i \wedge {}_l\llbracket \mathcal{G}\varphi \rrbracket_k^{\text{succ}(i)}$$



Encoding of loops - Eventually

- Given: LTL formula φ and path π with (k, l) -loop
- Recursive translation over the sub-terms of φ and states in π
- Introduce intermediate formula of the form ${}_l\llbracket \cdot \rrbracket_k^i$
 - l start-state of the loop
 - k bound
 - i current position
- Translation rule for $\mathcal{F}\varphi$:

$${}_l\llbracket \mathcal{F}\varphi \rrbracket_k^i := {}_l\llbracket \varphi \rrbracket_k^i \vee {}_l\llbracket \mathcal{F}\varphi \rrbracket_k^{\text{succ}(i)}$$



Encoding of loops

$$\begin{aligned}{}_I\llbracket p \rrbracket_k^i &:= p(s_i) \\{}_I\llbracket \neg p \rrbracket_k^i &:= \neg p(s_i) \\{}_I\llbracket \varphi \vee \psi \rrbracket_k^i &:= {}_I\llbracket \varphi \rrbracket_k^i \vee {}_I\llbracket \psi \rrbracket_k^i \\{}_I\llbracket \varphi \wedge \psi \rrbracket_k^i &:= {}_I\llbracket \varphi \rrbracket_k^i \wedge {}_I\llbracket \psi \rrbracket_k^i \\{}_I\llbracket \mathcal{G}\varphi \rrbracket_k^i &:= {}_I\llbracket \varphi \rrbracket_k^i \wedge {}_I\llbracket \mathcal{G}\varphi \rrbracket_k^{\text{succ}(i)} \\{}_I\llbracket \mathcal{F}\varphi \rrbracket_k^i &:= {}_I\llbracket \varphi \rrbracket_k^i \vee {}_I\llbracket \mathcal{F}\varphi \rrbracket_k^{\text{succ}(i)} \\{}_I\llbracket \varphi \mathcal{U} \psi \rrbracket_k^i &:= {}_I\llbracket \psi \rrbracket_k^i \vee ({}_I\llbracket \varphi \rrbracket_k^i \wedge {}_I\llbracket \varphi \mathcal{U} \psi \rrbracket_k^{\text{succ}(i)}) \\{}_I\llbracket \mathcal{X}\varphi \rrbracket_k^i &:= {}_I\llbracket \varphi \rrbracket_k^{\text{succ}(i)}\end{aligned}$$

Encoding without loops - Always

- Given: LTL formula φ and path π without (k, l) -loop
- Special case of loop translation
- Extension to infinite path with considering all properties beyond s_k as **false**
 - k bound
 - i current position
- Translation rule for $\mathcal{G}\varphi$, $i \leq k$:

$$\llbracket \mathcal{G}\varphi \rrbracket_k^i := \llbracket \varphi \rrbracket_k^i \vee \llbracket \mathcal{G}\varphi \rrbracket_k^{\text{succ}(i)}$$

Encoding without loops - Eventually

- Given: LTL formula φ and path π without (k, l) -loop
- Special case of loop translation
- Extension to infinite path with considering all properties beyond s_k as **false**
 - k bound
 - i current position
- Translation rule for $\mathcal{F}\varphi$, $i \leq k$:

$$\llbracket \mathcal{F}\varphi \rrbracket_k^i := \llbracket \varphi \rrbracket_k^i \vee \llbracket \mathcal{F}\varphi \rrbracket_k^{\text{succ}(i)}$$

Encoding without loops

$$\begin{aligned}\llbracket p \rrbracket_k^i &:= p(s_i) \\ \llbracket \neg p \rrbracket_k^i &:= \neg p(s_i) \\ \llbracket \varphi \vee \psi \rrbracket_k^i &:= \llbracket \varphi \rrbracket_k^i \vee \llbracket \psi \rrbracket_k^i \\ \llbracket \varphi \wedge \psi \rrbracket_k^i &:= \llbracket \varphi \rrbracket_k^i \wedge \llbracket \psi \rrbracket_k^i \\ \llbracket \mathcal{G}\varphi \rrbracket_k^i &:= \llbracket \varphi \rrbracket_k^i \wedge \llbracket \mathcal{G}\varphi \rrbracket_k^{i+1} \\ \llbracket \mathcal{F}\varphi \rrbracket_k^i &:= \llbracket \varphi \rrbracket_k^i \vee \llbracket \mathcal{F}\varphi \rrbracket_k^{i+1} \\ \llbracket \varphi \mathcal{U} \psi \rrbracket_k^i &:= \llbracket \psi \rrbracket_k^i \vee (\llbracket \varphi \rrbracket_k^i \wedge \llbracket \varphi \mathcal{U} \psi \rrbracket_k^{i+1}) \\ \llbracket \mathcal{X}\varphi \rrbracket_k^i &:= \llbracket \varphi \rrbracket_k^{i+1}\end{aligned}$$

General Translation to SAT-Problem

- Combining the components, BMC is encoded in propositional logic
- Given: LTL formula φ , Kripke structure M , bound k

$$\llbracket M, \varphi \rrbracket_k := \llbracket M \rrbracket_k \wedge \left((\neg L_k \wedge \llbracket \varphi \rrbracket_k^0) \vee \bigvee_{l=0}^k (T(s_k, s_l) \wedge \llbracket \varphi \rrbracket_k^l) \right)$$

- Unfolding of the transition relation
- There is no back loop \rightsquigarrow Translation without loops
- All possible starting points of a loop are considered \rightsquigarrow Translation for (k, l) -loop together with loop condition

Theorem

$\llbracket M, \varphi \rrbracket_k$ is satisfiable iff $M \models_k \mathbf{E}\varphi$

BMC is not complete

- Application: Start with $k = 0$ and increment until witness is found
- Termination is guaranteed iff witness exists ($M \models E\varphi$)
- If no witness exists, procedure does not terminate ($M \not\models E\varphi$)
- Upper bound for k to ensure property: Completeness threshold

Completeness Threshold

- For each (finite state) system M , property p and given translation scheme there exists a number \mathcal{CT} , called **completeness threshold**.
- Considering $\mathcal{G}\varphi$ formulas, \mathcal{CT} is equal to the **reachability diameter**, i.e., the minimal distance required to reach all (reachable) states of the system.

Definition (Reachability Diameter)

$$rd(M) := \min \left\{ i \mid \forall n > i. \forall s_0, \dots, s_n. \exists t \leq i. \exists s'_0, \dots, s'_t. \right.$$

$$\left. \left(I(s_0) \wedge \bigwedge_{j=0}^{n-1} T(s_j, s_{j+1}) \right) \rightarrow \left(I(s'_0) \wedge \bigwedge_{j=0}^{t-1} T(s'_j, s'_{j+1}) \wedge s'_t = s_n \right) \right\}$$

“Every state that is reachable in n steps, is also reachable in i steps.”

- This yields maximal shortest paths in the system.

Completeness Threshold

- Problem: One has to choose n
- Let V be the set of variables defining the states. Worst case: $n = 2^{|V|}$
- Better: Choose $n = i + 1$.

Definition (Reachability Diameter)

$$rd(M) := \min \left\{ i \mid \forall s_0, \dots, s_{i+1}. \exists s'_0, \dots, s'_i. \right.$$

$$\left. \left(I(s_0) \wedge \bigwedge_{j=0}^i T(s_j, s_{j+1}) \right) \rightarrow \left(I(s'_0) \wedge \bigwedge_{j=0}^{i-1} T(s'_j, s'_{j+1}) \wedge \bigvee_{j=0}^i s'_j = s_{i+1} \right) \right\}$$

“Every state that is reachable in $i + 1$ steps, it is also reachable in i steps.”

Completeness Threshold

- Problem: Formula contains alternation of quantifiers
- Solution: Over-approximation of $rd(M)$

Definition (Recurrence Diameter)

$rdr(M) :=$

$$\max \left\{ i \mid \exists s_0 \dots s_i : I(s_0) \wedge \bigwedge_{j=0}^{i-1} T(s_j, s_{j+1}) \wedge \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^i s_j \neq s_k \right\}$$

“Longest loop-free initial path in M .”

- As every shortest path is a loop-free path, this is an over-approximation of $rd(M)$.