Timing Is Everything

Joël Ouaknine

Department of Computer Science
Oxford University

BCS Meeting, Oxford

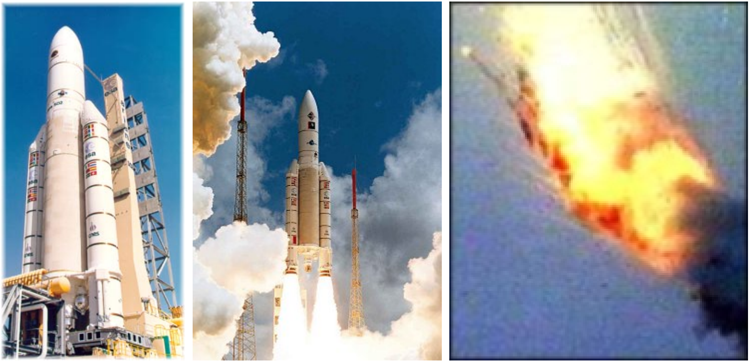
17 May 2012

Automated Verification

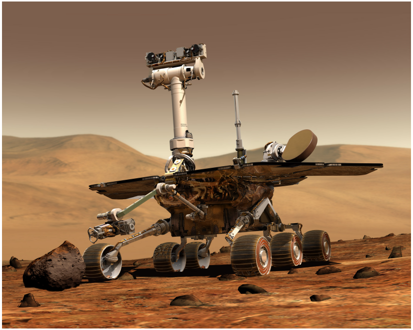
“In theory, there is no difference between theory and practice. In practice, there is.”

Jan L.A. van de Snepscheut

Ariane 5 Explosion, French Guyana, 1996

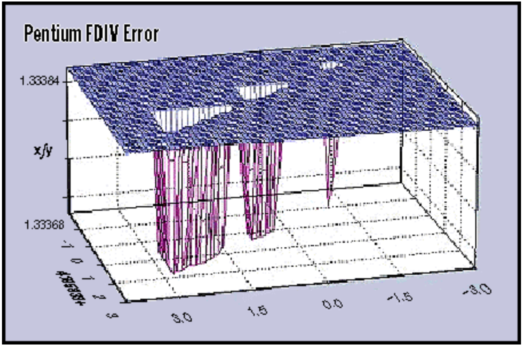


NASA Mars Missions, 1997–2004

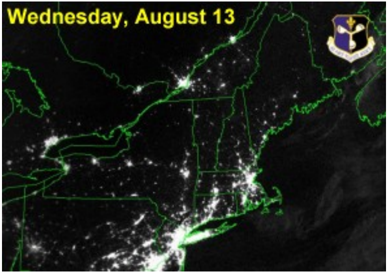


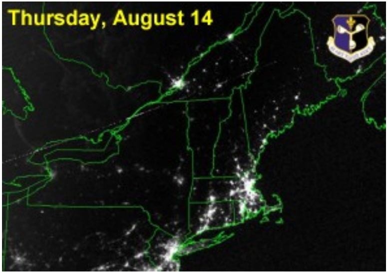
- 1997: Mars Rover loses contact
- 1999: Mars Climate Orbiter is lost
- 1999: Mars Polar Lander is lost
- 2004: Mars Rover freezes

Intel Pentium FDIV Bug, 1994



Northeast Blackout, 2003





Chrysler Pacifica SUV, 2006

December 2006: DaimlerChrysler recalls 128,000 Pacifica sports utility vehicles because of a problem with the software governing the fuel pump and power train control. The defect could cause the engine to stall unexpectedly. [Washington Post]

Automated Verification

"A Grand Challenge for computing research."

Sir Tony Hoare, 2003

Automated Verification

"A Grand Challenge for computing research."

Sir Tony Hoare, 2003

Now one of a small handful of areas *'targetted for growth'* by UK funding council EPSRC.

Automated Verification

“Nobody is going to run into a friend’s office with a program verification. Nobody is going to sketch a verification out on a paper napkin. . . One can feel one’s eyes glaze over at the very thought.”

Rich de Millo, Richard Lipton, Alan Perlis, 1979

Automated Verification

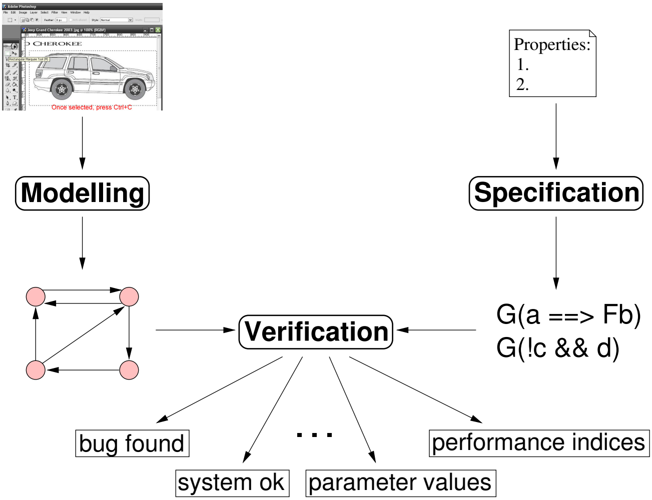
"Nobody is going to run into a friend's office with a program verification. Nobody is going to sketch a verification out on a paper napkin. . . One can feel one's eyes glaze over at the very thought."

Rich de Millo, Richard Lipton, Alan Perlis, 1979

"The success of program verification as a generally applicable and completely reliable method for guaranteeing program performance is not even a theoretical possibility."

James H. Fetzer
Program Verification: The Very Idea, CACM 31(9), 1988

Automated Verification: A High-Level Overview



SLAM

TERMINATOR

proof tools for termination and liveness

TERMINATOR vs. The Ackermann Function

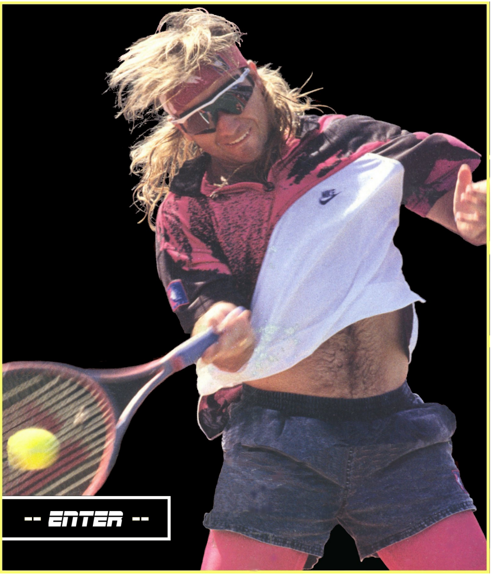
```
int Ack(int m, int n) {
    if (m == 0)
        return n + 1;
    else if (n == 0)
        return Ack(m - 1, 1);
    else
        return Ack(m - 1, Ack(m, n - 1));
}
```

TERMINATOR vs. The Ackermann Function

```
int Ack(int m, int n) {
    if (m == 0)
        return n + 1;
    else if (n == 0)
        return Ack(m - 1, 1);
    else
        return Ack(m - 1, Ack(m, n - 1));
}
```

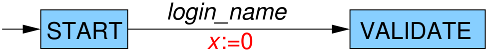
$$\mathrm{Ack}(n,n):\quad 1,\quad 3,\quad 7,\quad 61,\quad 2^{2^{2^{2^{2^{2^{2}}}}}}-3,\quad \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{\mathrm{Ack}(5,4)+3}-3$$

Timing Is Everything

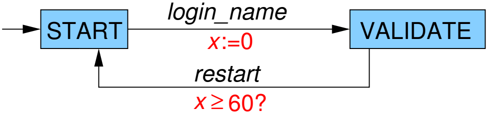


A Login Protocol

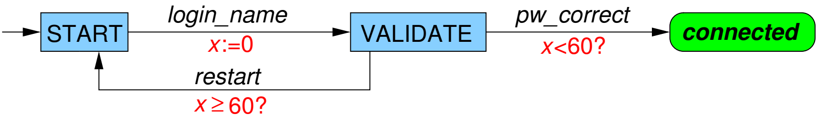
A Login Protocol



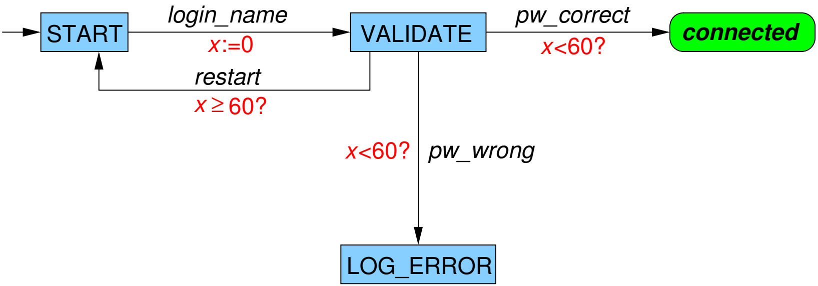
A Login Protocol



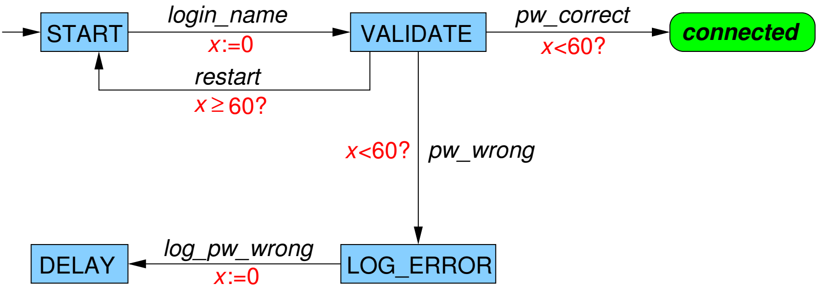
A Login Protocol



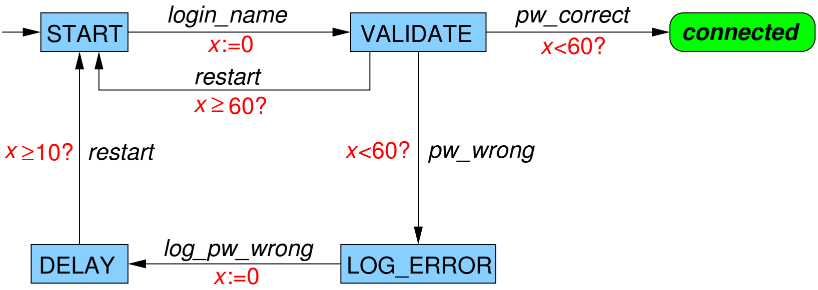
A Login Protocol



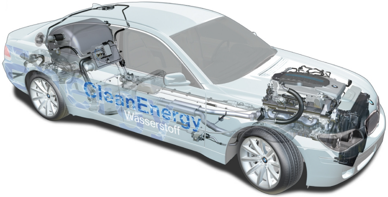
A Login Protocol



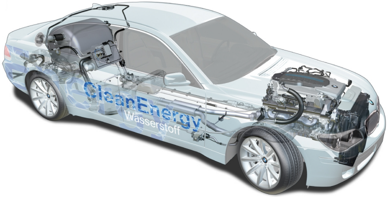
A Login Protocol

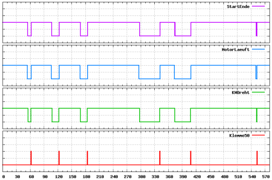


BMW Hydrogen 7

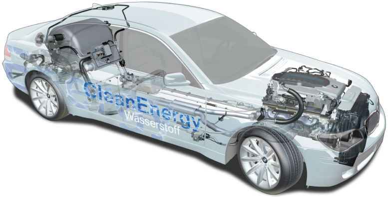


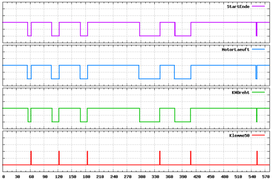
BMW Hydrogen 7





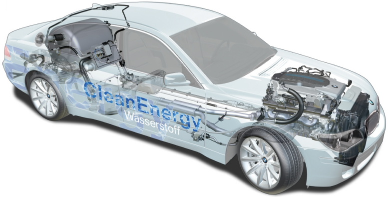
BMW Hydrogen 7

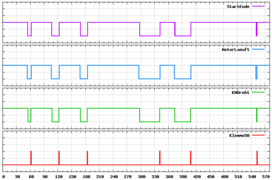




$$\Box(\mathit{PEDAL} \rightarrow \Diamond\, \mathit{BRAKE})$$

BMW Hydrogen 7





$$\square(PEDAL \rightarrow \Diamond\, BRAKE)$$

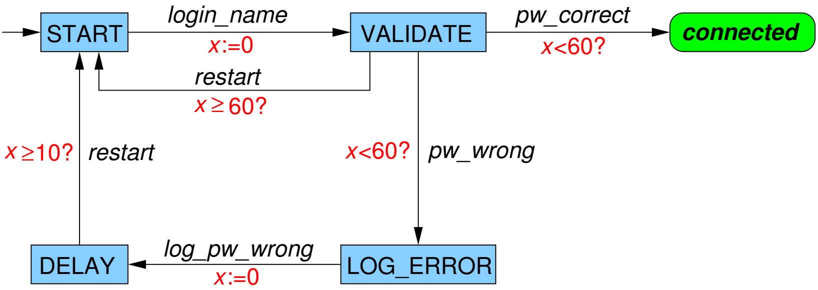
$$\square(PEDAL \rightarrow \Diamond_{[25,40]}\, BRAKE)$$

Timed Automata

Introduced by Rajeev Alur at Stanford during his PhD under David Dill:

- Rajeev Alur, David L. Dill: *Automata For Modeling Real-Time Systems*. ICALP 1990: 322-335
- Rajeev Alur, David L. Dill: *A Theory of Timed Automata*. TCS 126(2): 183-235, 1994

Timed Automata



Timed Automata

Time is modelled as the non-negative reals, $\mathbb{R}_{\geq 0}$.

Timed Automata

Time is modelled as the non-negative reals, $\mathbb{R}_{\geq 0}$.

Theorem (Alur, Courcourbetis, Dill 1990)

Reachability is decidable (in fact PSPACE-complete).

Timed Automata

Time is modelled as the non-negative reals, $\mathbb{R}_{\geq 0}$.

Theorem (Alur, Courcourbetis, Dill 1990)

Reachability is decidable (in fact PSPACE-complete).

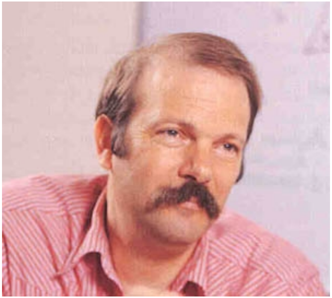
Unfortunately:

Theorem (Alur & Dill 1990)

Language inclusion is undecidable for timed automata.

Temporal Logic Model Checking

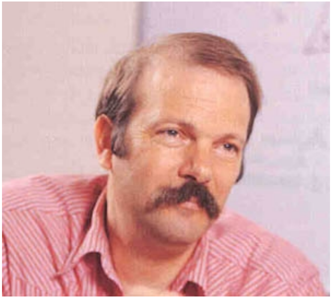
"The paradigmatic idea of the automata-theoretic approach to verification is that we can compile high-level logical specifications into an equivalent low-level finite-state formalism."



Moshe Vardi

Temporal Logic Model Checking

"The paradigmatic idea of the automata-theoretic approach to verification is that we can compile high-level logical specifications into an equivalent low-level finite-state formalism."

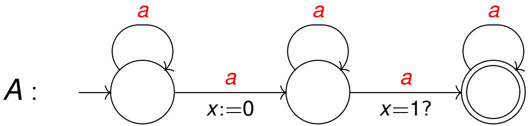


Moshe Vardi

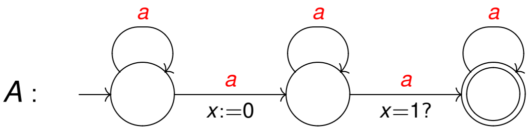
Theorem

Automata are closed under all Boolean operations. Moreover, the language inclusion problem [$L(A) \subseteq L(B)$?] is decidable.

An Uncomplementable Timed Automaton

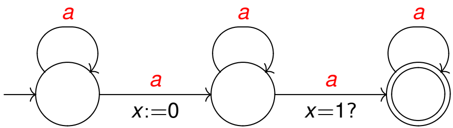


An Uncomplementable Timed Automaton



An Uncomplementable Timed Automaton

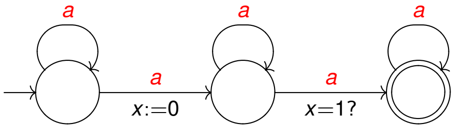
A :



L(A):

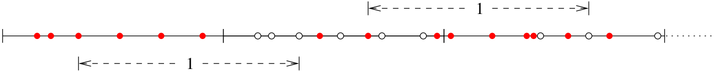
An Uncomplementable Timed Automaton

A :

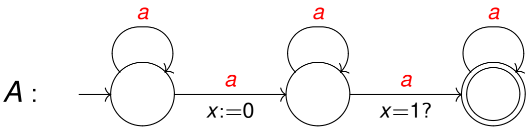


L(A):

$\overline{L(A)}$:

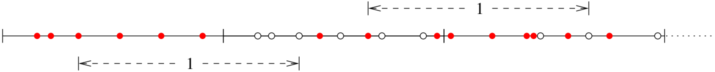


An Uncomplementable Timed Automaton



$L(A)$:

$\overline{L(A)}$:



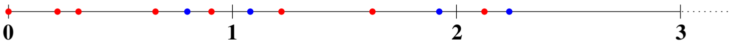
A cannot be complemented:
There is no timed automaton *B* with $L(B) = \overline{L(A)}$.

Metric Temporal Logic

$$\Box(a \rightarrow \Diamond_{[0,1]} b)$$

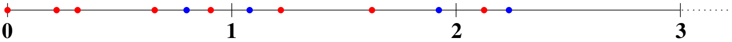
Metric Temporal Logic

$$\Box(a \rightarrow \Diamond_{[0,1]}\, b)$$



Metric Temporal Logic

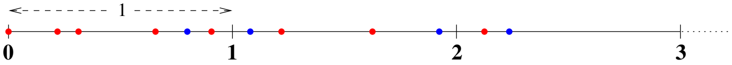
$$\Box(a \rightarrow \Diamond_{[0,1]}\, b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

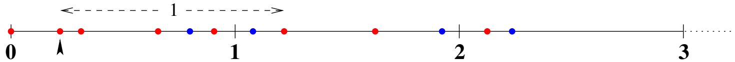
$$\Box(a \rightarrow \Diamond_{[0,1]}\, b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

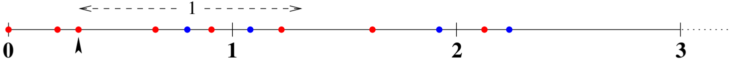
$$\Box(a \rightarrow \Diamond_{[0,1]}\, b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

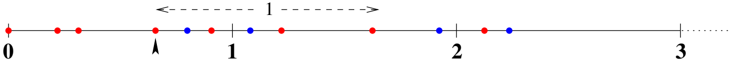
$$\Box(a \rightarrow \Diamond_{[0,1]}\, b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

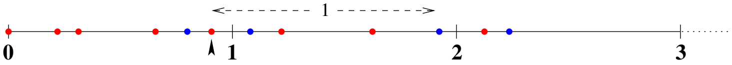
$$\Box(a \to \Diamond_{[0,1]}\, b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

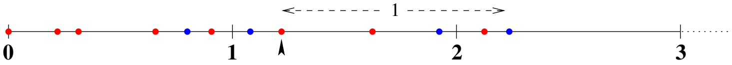
$$\Box(a \rightarrow \Diamond_{[0,1]}\, b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

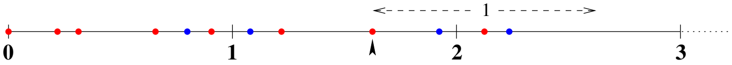
$$\Box(a \to \Diamond_{[0,1]} b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

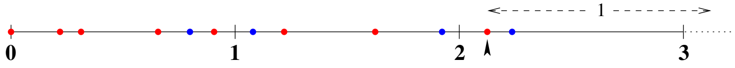
$$\Box(a \rightarrow \Diamond_{[0,1]}\, b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

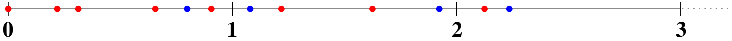
$$\Box(a \rightarrow \Diamond_{[0,1]} b)$$



Does the timed word satisfy the specification?

Metric Temporal Logic

$$\Box(a \rightarrow \Diamond_{[0,1]}\, b)$$



Does the timed word satisfy the specification? Yes.

Real-Time Model Checking

Given a timed automaton A and a Metric Temporal Logic specification φ, does every timed word of A satisfy φ?

Real-Time Model Checking

Given a timed automaton A and a Metric Temporal Logic specification φ, does every timed word of A satisfy φ?

- For about 15 years ($\sim$ 1990–2005), the real-time model-checking problem was widely claimed in the literature to be undecidable.

Real-Time Model Checking

Given a timed automaton A and a Metric Temporal Logic specification φ, does every timed word of A satisfy φ?

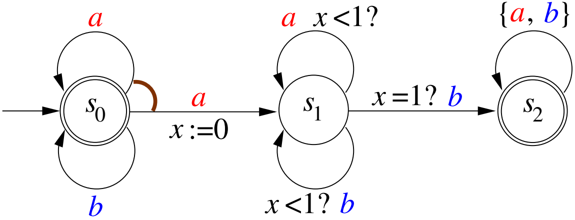
- For about 15 years ($\sim$ 1990–2005), the real-time model-checking problem was widely claimed in the literature to be undecidable.
- In 2005, James Worrell and I showed decidability through the development of the theory of timed alternating automata.


FBI
FBI

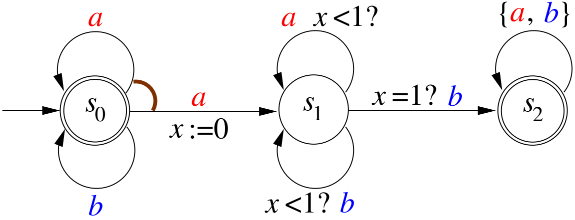

NICOLAS CAGE
JULIANNE MOORE
JESSICA BIEL
NEXT
"A CLEVER, ACTION-PACKED RIDE!"

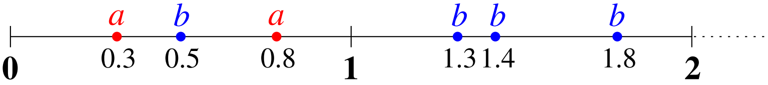
$$\Box(a \rightarrow \Diamond_{=1} b)$$

$\Box(a \rightarrow \Diamond_{=1} b)$

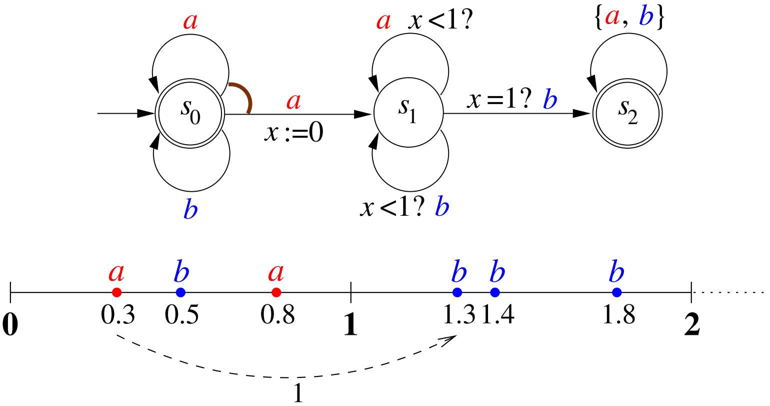


$$\Box(a \rightarrow \Diamond_{=1} b)$$

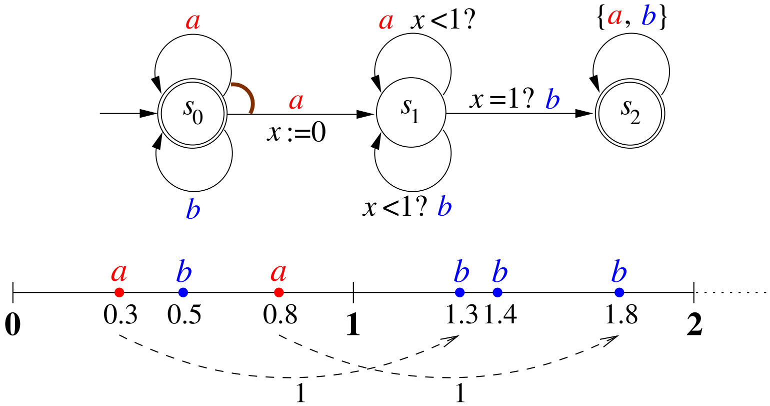




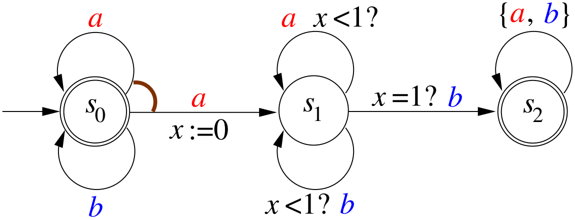
$\Box(a \rightarrow \Diamond_{=1}\, b)$

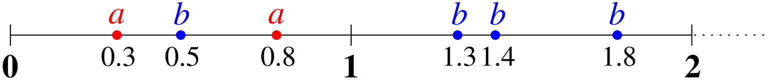


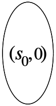
$\Box(a \rightarrow \Diamond_{=1} b)$



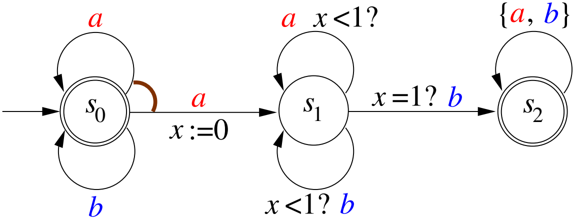
$\Box(a \to \Diamond_{=1}\, b)$

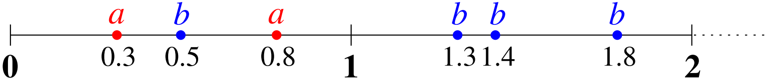


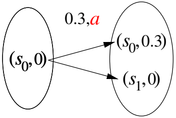




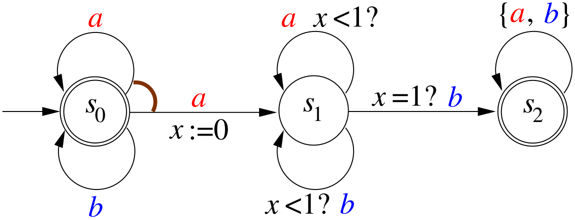
$$\Box(a \rightarrow \Diamond_{=1} b)$$

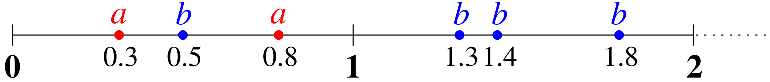


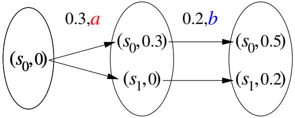




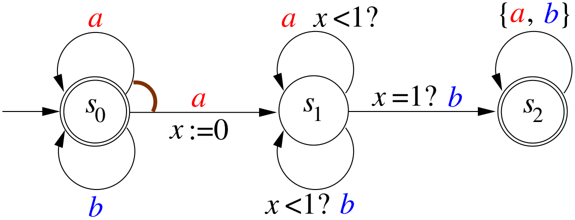
$$\Box(a \rightarrow \Diamond_{=1} b)$$

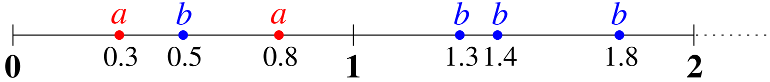


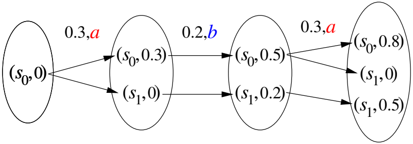




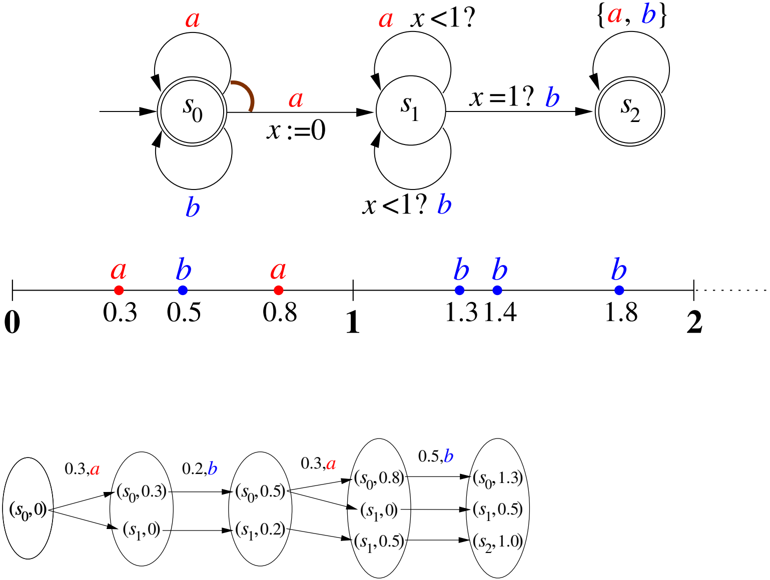
$$\Box(a \rightarrow \Diamond_{=1} b)$$



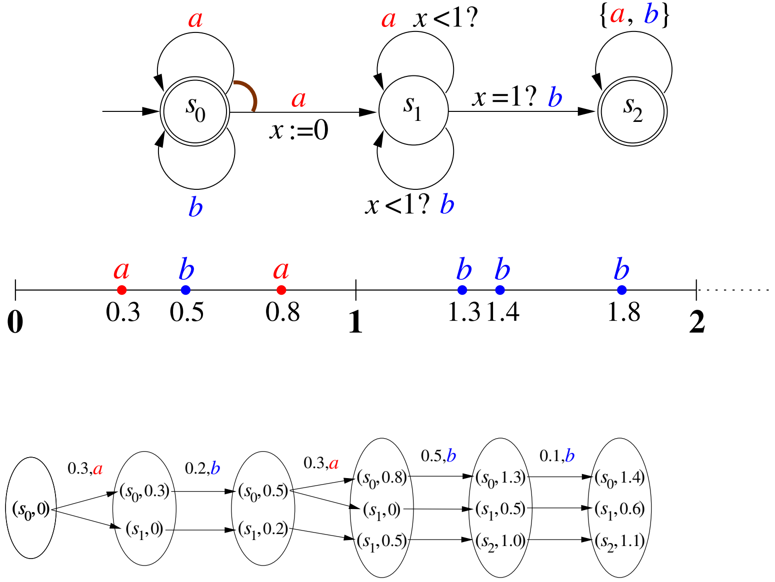




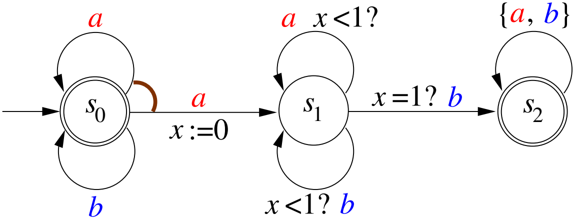
$$\Box(a \rightarrow \Diamond_{=1} b)$$

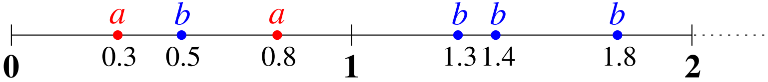


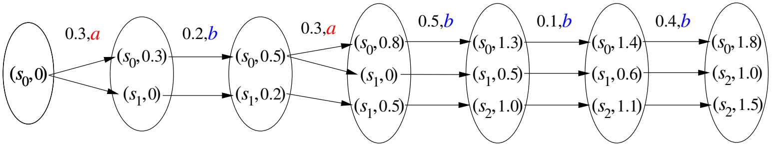
$\Box(a \rightarrow \Diamond_{=1} b)$



$$\Box(a \rightarrow \Diamond_{=1} b)$$







Real-Time Model Checking: A High-Level Algorithm

Real-time model checking problem

Real-Time Model Checking: A High-Level Algorithm

Real-time model checking problem

$\Downarrow$

Alternating timed automaton emptiness problem

Real-Time Model Checking: A High-Level Algorithm

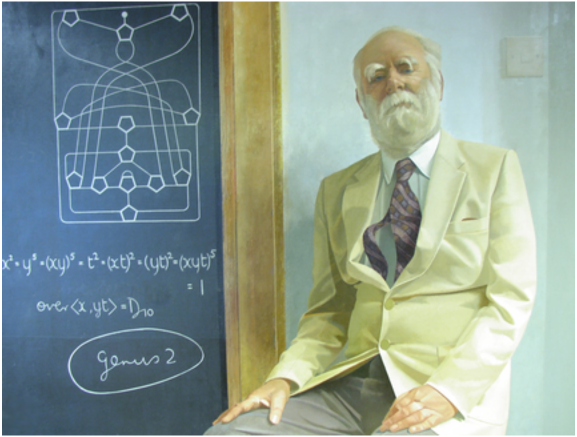
Real-time model checking problem

$\Downarrow$

Alternating timed automaton emptiness problem

$\Downarrow$

Halting problem for Turing machine with insertion errors


x²·y⁵·(xy)⁵·t²·(xt)²·(yt)²·(xyt)⁵
= 1
over ⟨x, yt⟩ = D10
genus 2

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

Higman's Lemma

Theorem
The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab, aa

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab, aa, ba

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab, aa, ba, bbb

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab, aa, ba, bbb, abb

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab, aa, ba, bbb, abb, ab

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab, aa, ba, bbb, abb, ab, a

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab, aa, ba, bbb, abb, ab, a, bb

Higman's Lemma

Theorem

The subword order over a finite alphabet is a well-quasi order.

(Graham Higman, *Ordering by Divisibility in Abstract Algebras*, Proceedings of the London Mathematical Society, vol. 2, 1952.)

"HIGMAN" is a subword of "HIGHMOUNTAIN".

Any infinite sequence of words $W_1, W_2, W_3, \ldots$ must eventually have two words, W_i and W_{i+k}, such that the first is a subword of the second.

- aba, abbb, baab, aa, ba, bbb, abb, ab, a, bb, b

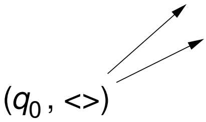
The Halting Problem for Faulty Turing Machines

$(q_0 , \text{<>})$

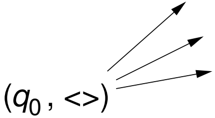
The Halting Problem for Faulty Turing Machines

$(q_0, <>)$

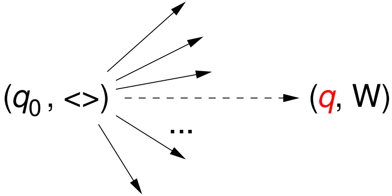
The Halting Problem for Faulty Turing Machines



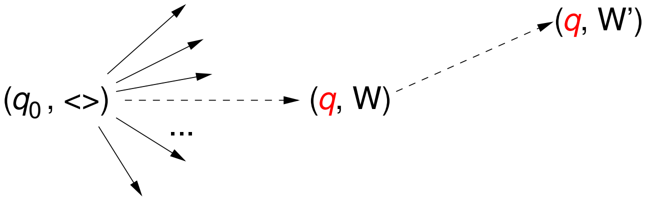
The Halting Problem for Faulty Turing Machines



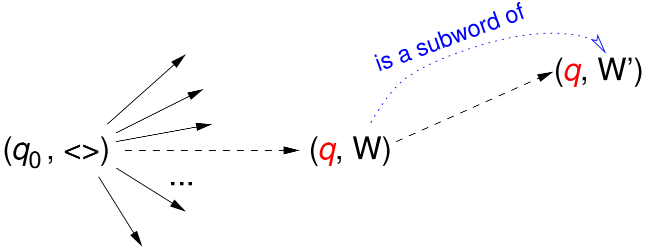
The Halting Problem for Faulty Turing Machines



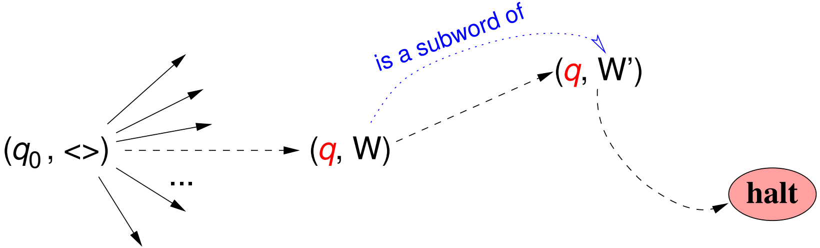
The Halting Problem for Faulty Turing Machines



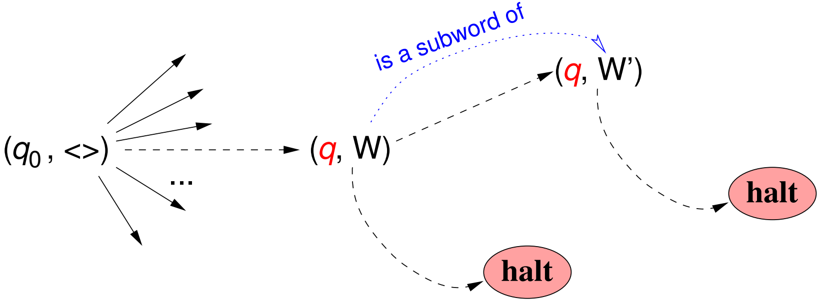
The Halting Problem for Faulty Turing Machines



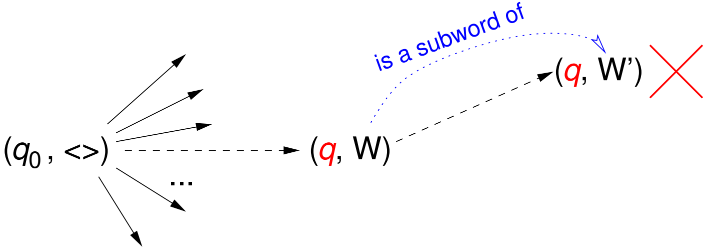
The Halting Problem for Faulty Turing Machines



The Halting Problem for Faulty Turing Machines



The Halting Problem for Faulty Turing Machines



Real-Time Model Checking

Theorem

The real-time model-checking problem for Metric Temporal Logic is decidable (under the pointwise semantics).

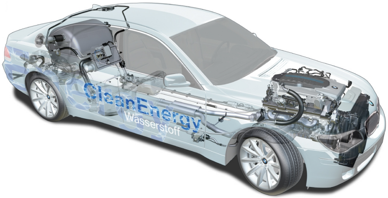
Real-Time Model Checking

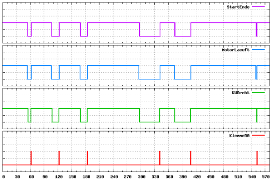
Theorem

The real-time model-checking problem for Metric Temporal Logic is decidable (under the pointwise semantics).

The complexity is provably non-primitive recursive. In particular, it grows faster than Ackermann's function in the worst case.

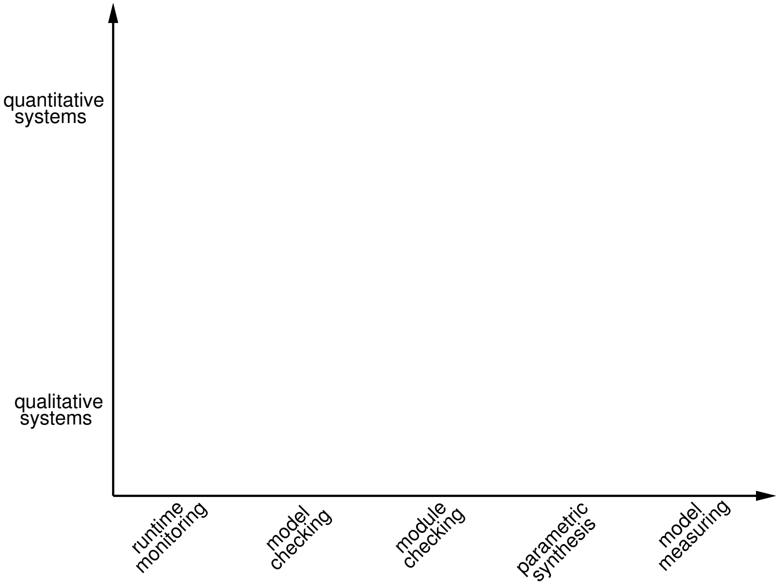
From Timed Alternating Automata to Efficient Runtime Monitoring Algorithms



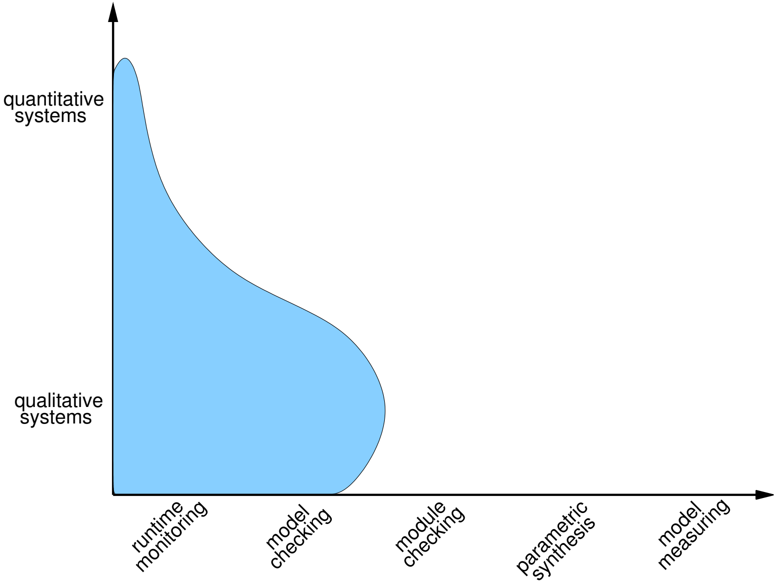


$$\Box(\mathit{PEDAL} \rightarrow \Diamond_{[25,40]}\ \mathit{BRAKE})$$

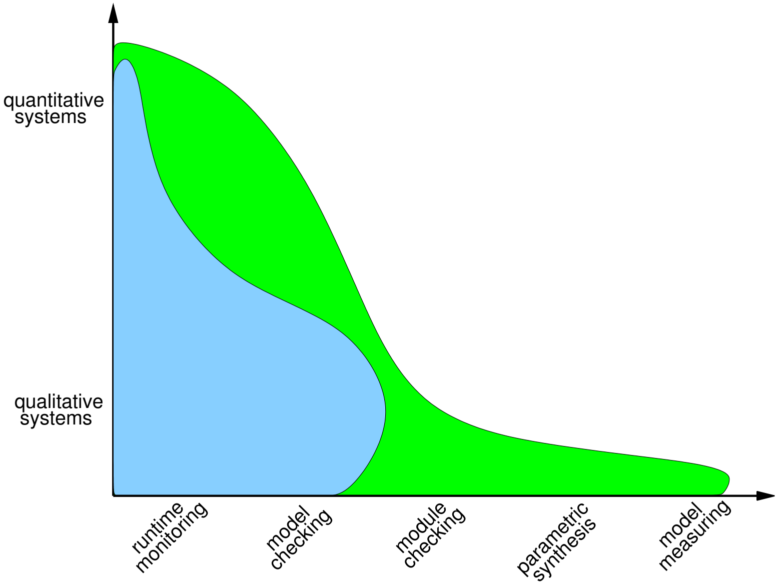
Quantitative Verification: From Model Checking to Model Measuring



Quantitative Verification:
From Model Checking to Model Measuring



Quantitative Verification:
From Model Checking to Model Measuring



Quantitative Verification:
From Model Checking to Model Measuring