Over Betrouwbaarheid van Computersystemen

Frits Vaandrager

Institute for Computing and Information Sciences
Radboud Universiteit Nijmegen

Betrouwbaarheid Computersystemen

Onze samenleving is in hoge mate afhankelijk van computersystemen

Is ons vertrouwen in deze systemen gegrond?

Hoe kunnen we betrouwbare systemen bouwen?

Sint Servaasbrug Maastricht

“Uptime”:

ruim 700 jaar

Ariane 5 Raket

“Uptime”:
40 sec

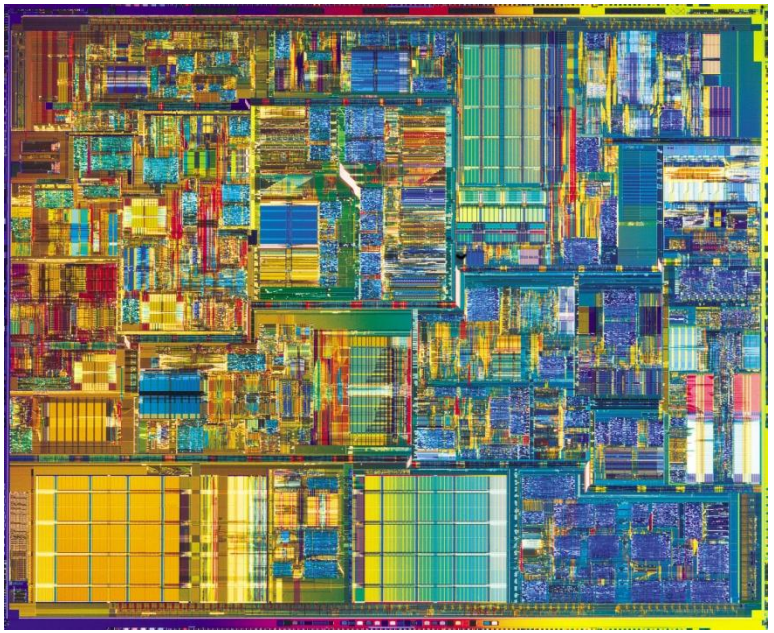
Aandeel ICT in productiekosten auto's

- 2000: 26%
- 2010: 48%

Software is absoluut het meest complexe artefact dat de mens routinematig bouwt…

Tussen 10^{69} en 10^{81} atomen in het universum

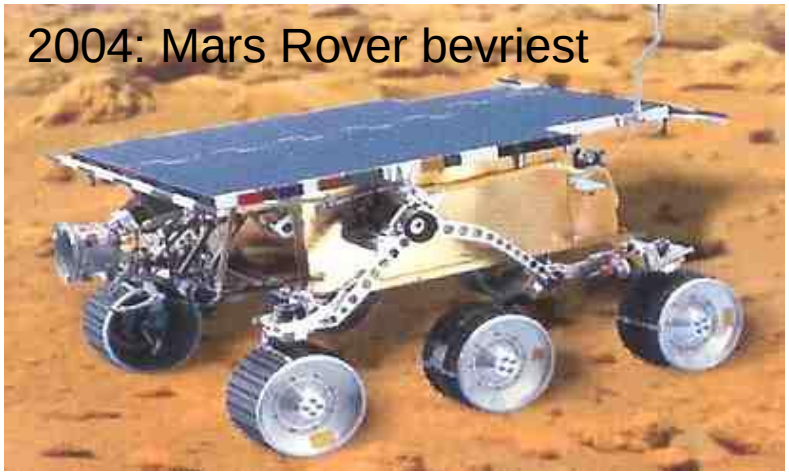
10 MB geheugen > $10^{20.000.000}$ toestanden



Software is niet continu: wijziging van 1 bit in een programma kan leiden tot volstrekt ander gedrag!

Geen verrassing dus dat het vrijwel nooit foutloos werkt!

2004: Mars Rover bevriest

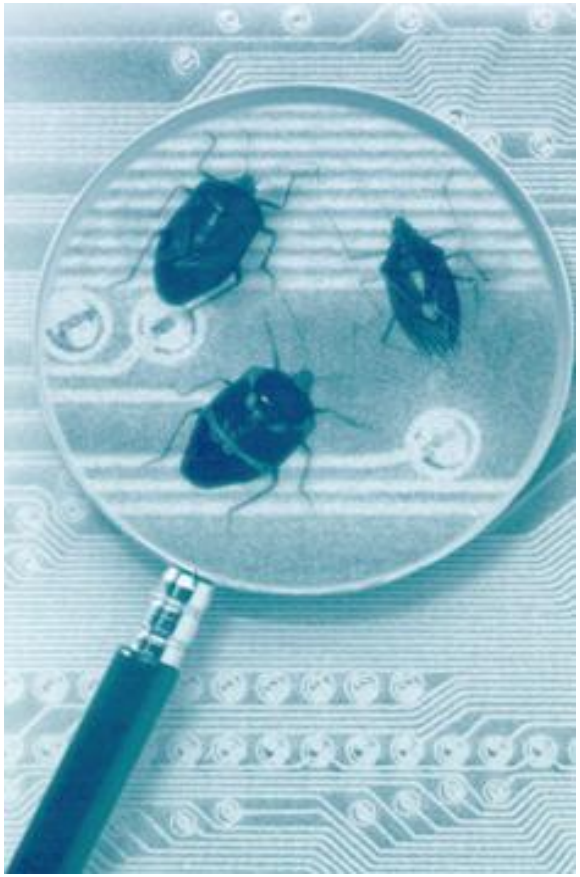


2005: Computer “kaapt” vliegtuig

2006: DaimlerChrysler roept 128.000 Pacifica auto’s terug

2007: Computer crashes bij ProRail

Onderzoek Radboud Universiteit



- **Het bouwen van modellen**

  Beschrijf relevante aspecten van systeem formeel (in wiskundige taal)
- **Model checking**

  Gebruik computer om alle toestanden van model te exploreren
- **Model-gebaseerd testen**

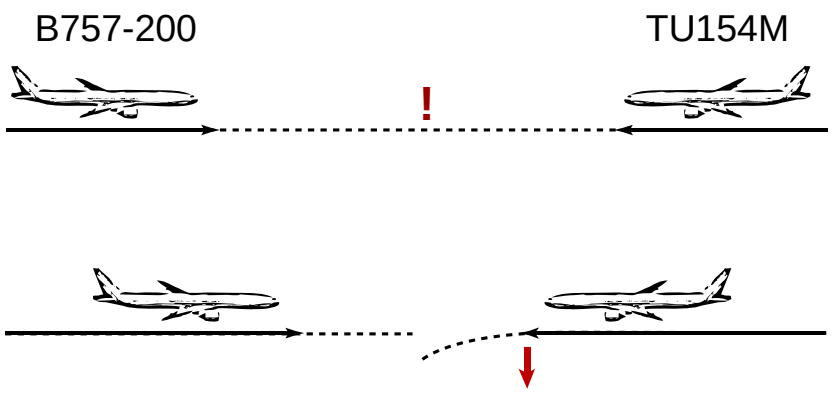
  Gebruik model om het systeem op goede manier te testen
- **Doel**

  Fouten (“bugs”) opsporen

Voorbeeld: Überlingen, 1 Juli 2002

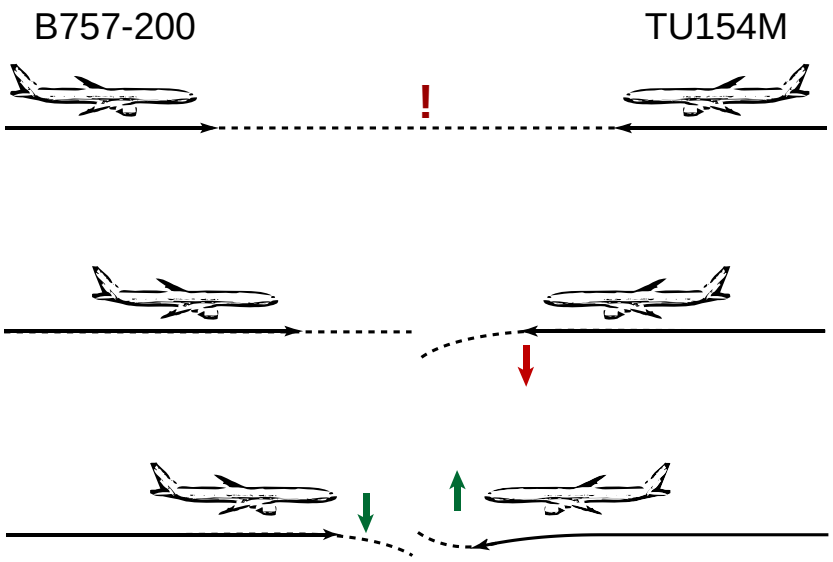
- **Boeing & Tupolew kruisen**
- **21:33:03**
  - Alarm door het Collision Avoidance System (TCAS)

Voorbeeld: Überlingen, 1 Juli 2002



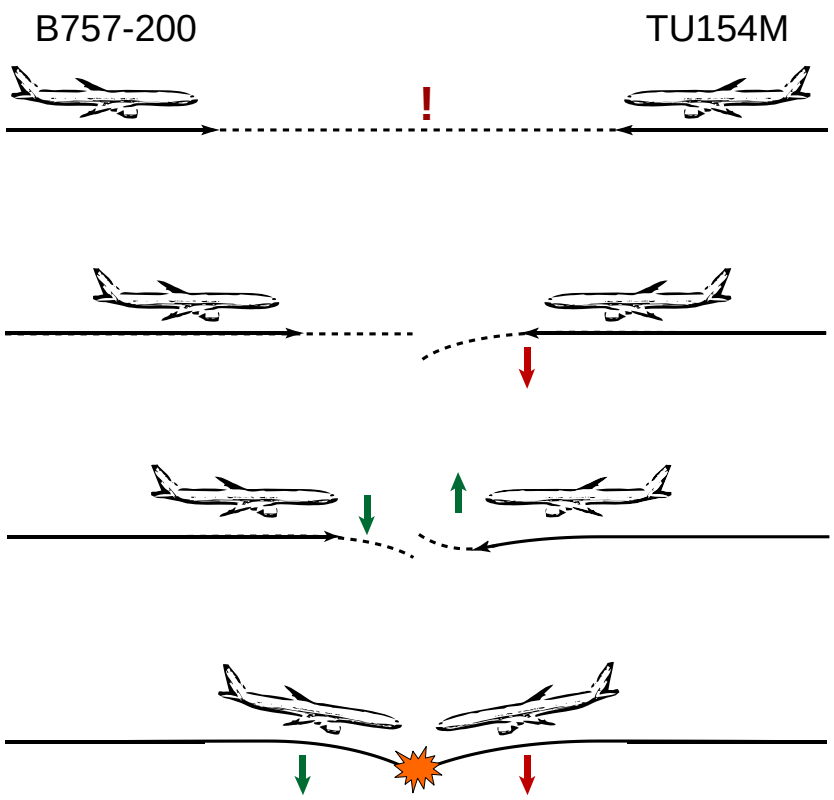
- **Boeing & Tupolew kruisen**
- **21:33:03**
  - Alarm door het Collision Avoidance System (TCAS)
- **21:34:49**
  - Opdracht verkeersleider

Voorbeeld: Überlingen, 1 Juli 2002



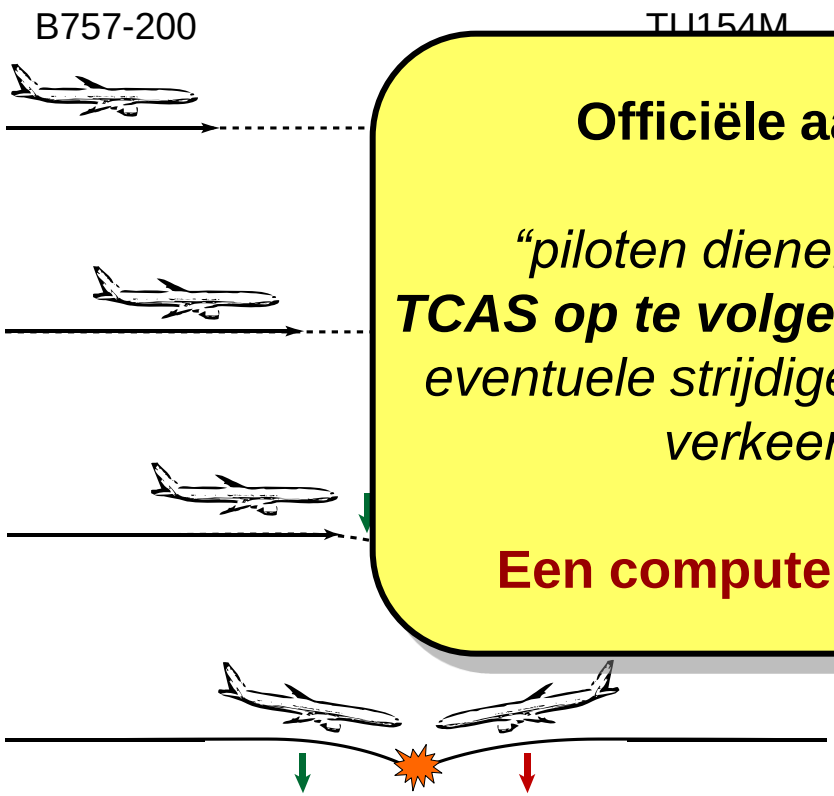
- **Boeing & Tupolew kruisen**
- **21:33:03**
  - Alarm door het Collision Avoidance System (TCAS)
- **21:34:49**
  - Opdracht verkeersleider
- **21:34:56**
  - TCAS aanbeveling

Voorbeeld: Überlingen, 1 Juli 2002



- **Boeing & Tupolew kruisen**
- **21:33:03**
  - Alarm door het Collision Avoidance System (TCAS)
- **21:34:49**
  - Opdracht verkeersleider
- **21:34:56**
  - TCAS aanbeveling
- **21:35:32**
  - Botsing

Voorbeeld: Überlingen, 1 Juli 2002



Officiële aanbeveling:

*"piloten dienen adviezen van **TCAS op te volgen**, onafhankelijk van eventuele strijdige adviezen door de verkeersleiding"*

Een computer vertrouwen!?

- **Boeing & Tupolew kruisen**

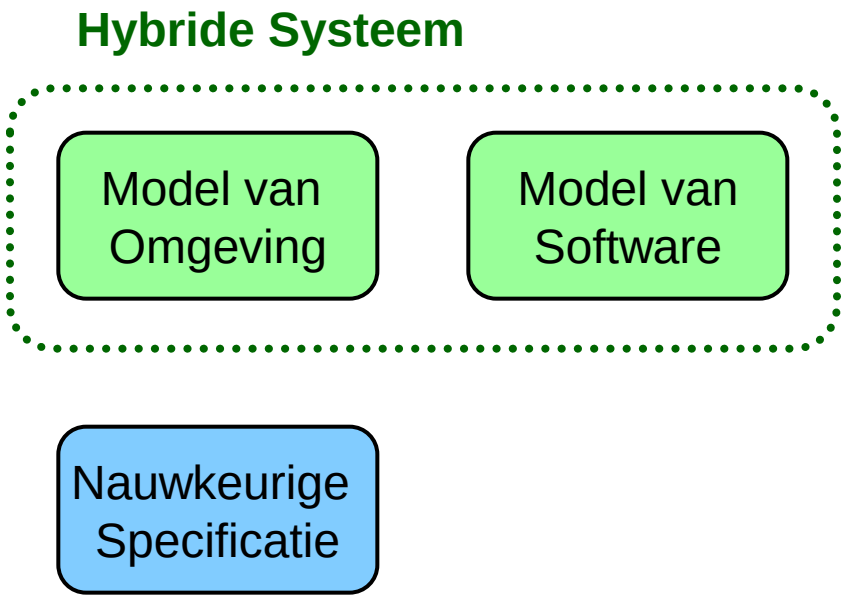
et Collision
stem (TCAS)

keersleider

veling

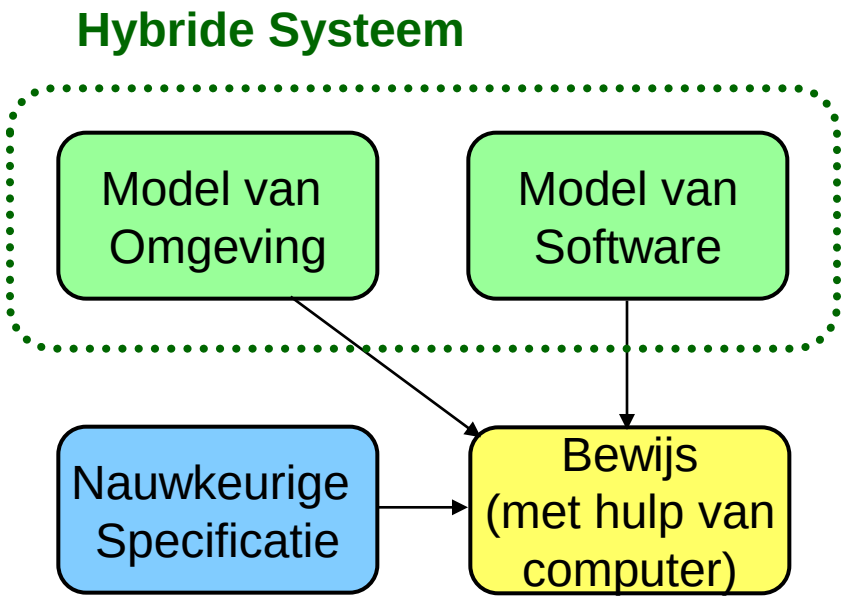
- **21:35:32**
  - Botsing

Formele Verificatie



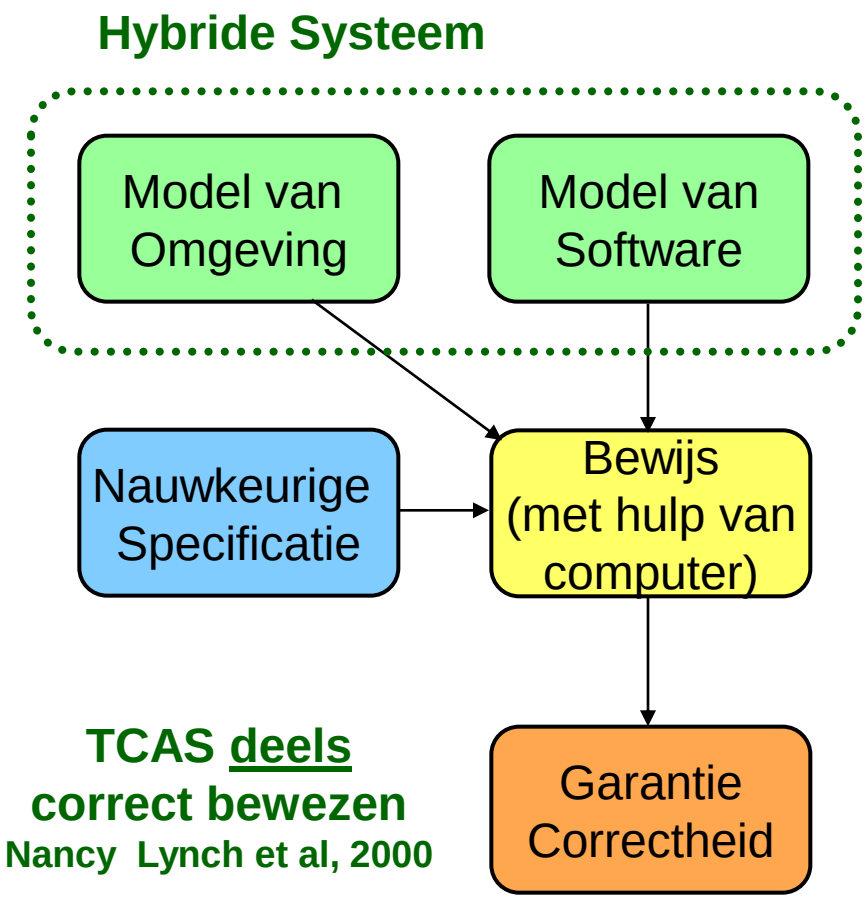
- **Kenmerken**
  - Gebruik taal van de wiskunde
  - Computerondersteuning
- **Hybride Systeem**
  - continue omgeving
  - discrete software

Formele Verificatie



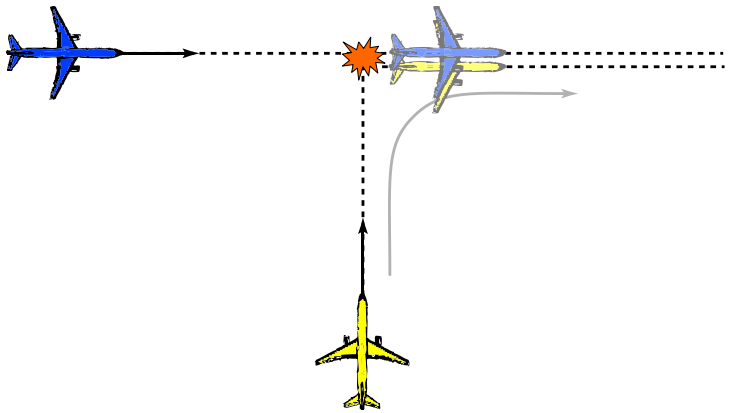
- **Kenmerken**
  - Gebruik taal van de wiskunde
  - Computerondersteuning
- **Hybride Systeem**
  - continue omgeving
  - discrete software
- **Problemen**
  - Lukt alleen voor eenvoudige modellen
  - Alle mogelijkheden moeten doorlopen worden
    $\Rightarrow$ toestandsexplosie
- **Oplossingen**
  - abstractie
  - compositionaliteit

Formele Verificatie



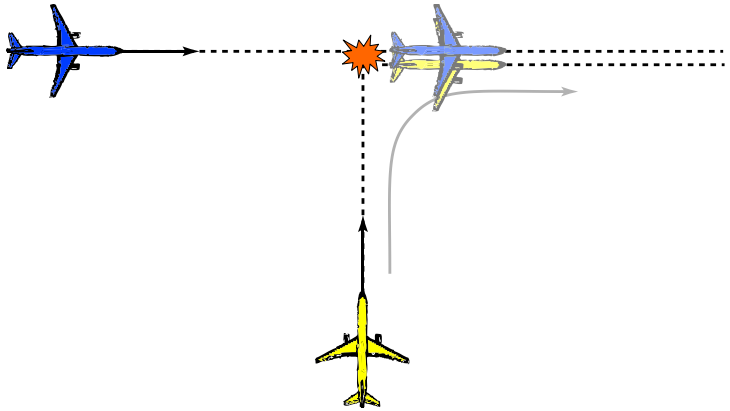
- **Kenmerken**
  - Gebruik taal van de wiskunde
  - Computerondersteuning
- **Hybride Systeem**
  - continue omgeving
  - discrete software
- **Problemen**
  - Lukt alleen voor eenvoudige modellen
  - Alle mogelijkheden moeten doorlopen worden
    $\Rightarrow$ toestandsexplosie
- **Oplossingen**
  - abstractie
  - compositionaliteit

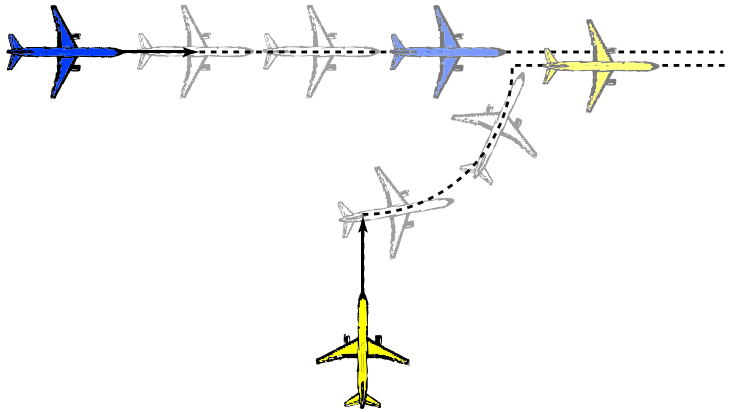
Voorbeeld: Invoegmanoeuvre [Frehse]



- **Probleem verkeersleiding**
  - Invoegen bij verschillende snelheden
- **Doelen**
  - Voorkom botsingen
  - Behoudt onderlinge afstand

Voorbeeld: Invoegmanoeuvre [Frehse]

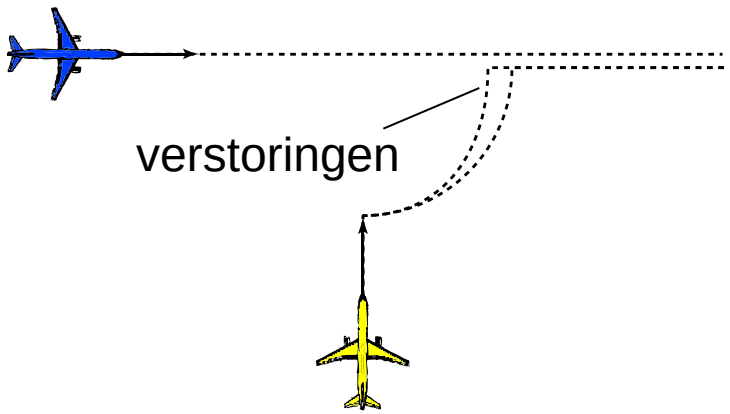




- **Probleem verkeersleiding**
  - Invoegen bij verschillende snelheden
- **Doelen**
  - Voorkom botsingen
  - Behoudt onderlinge afstand

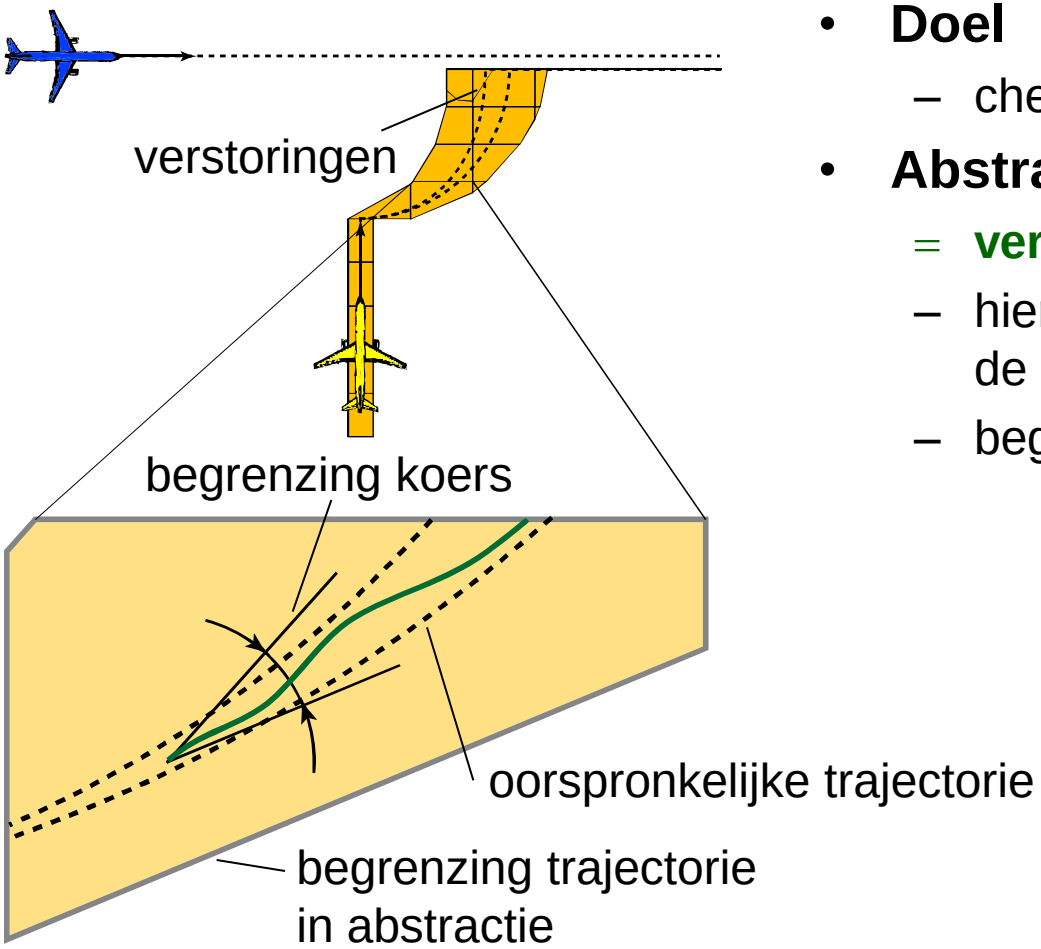
- **Model**
  - Omgeving: Vliegtuigen
  - Software: Besturing
    - snel/langzaam
- **Specificatie**
  - Behoudt minimale afstand

Abstractie



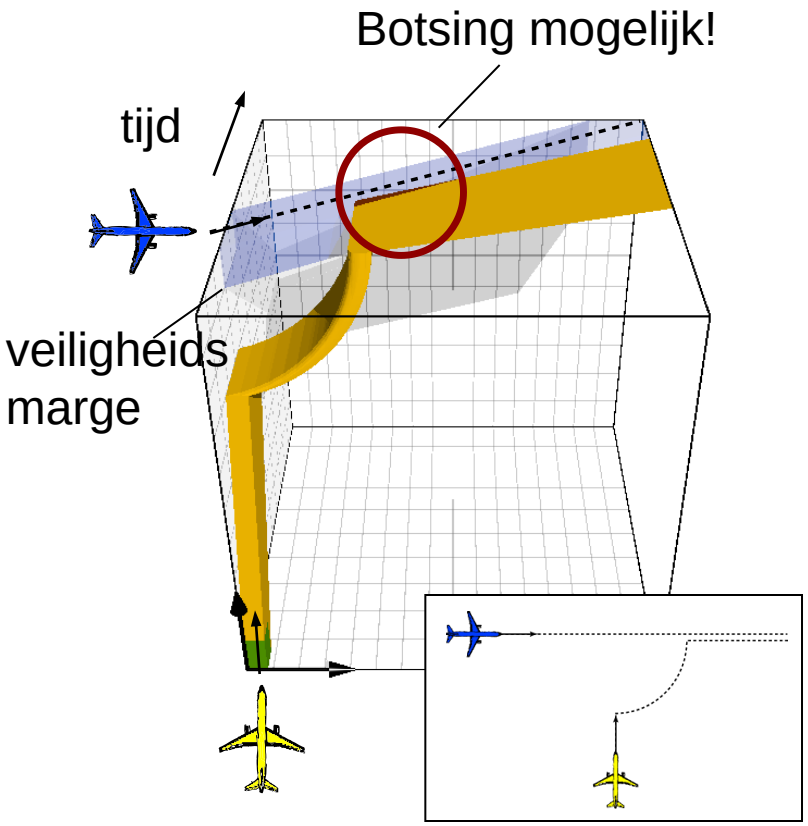
- **Doel**
  - check **alle** mogelijkheden
- **Abstractie**
  - = **vereenvoudigd model**
  - hier: lineaire afschattingen van de koers

Abstractie



- **Doel**
  - check **alle** mogelijkheden
- **Abstractie**
  - = **vereenvoudigd model**
  - hier: lineaire afschattingen van de koers
  - begrenzing trajectories

State-of-the-art



- **Grote vooruitgang op het gebied van formele verificatie**
- **Hoeveelheid software in vliegtuigen, treinen en auto's groeit nog veel sneller**
- **Complexiteit software wordt systematisch onderschat**
- **"Radical Design"**
- **Pas op met de allernieuwste (dure) auto's, treinen, en vliegtuigen**

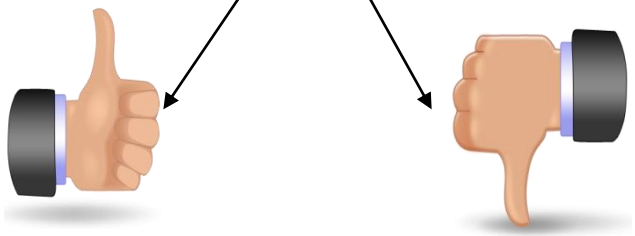
Model Checking

M: Traffic Light Controller

P: No Collisions

Model Checker

Does M satisfy P?



Yes!

No, and here's an example of why not.

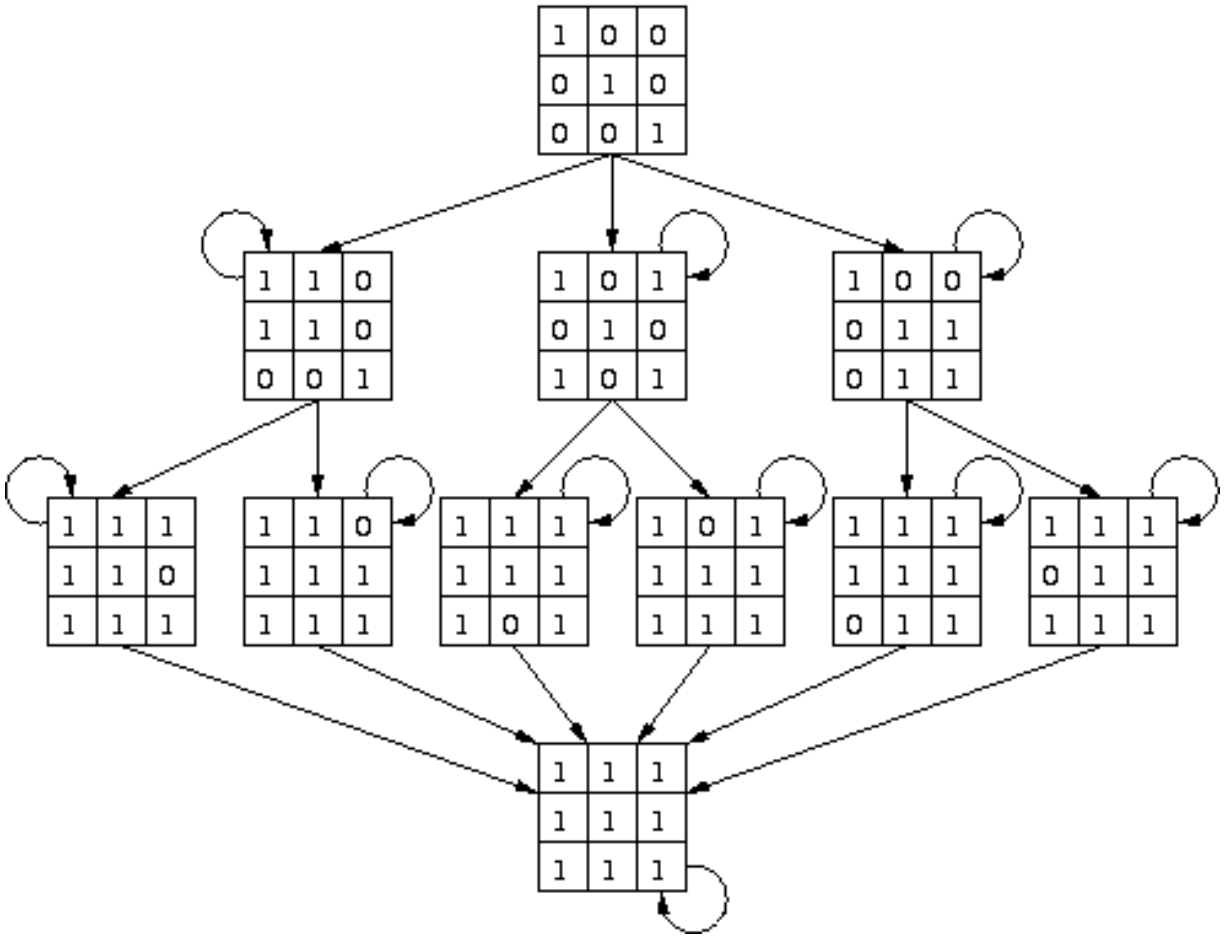
Demonstratie Model Checker

Zes vriendinnen hebben ieder een roddel. Ze bellen elkaar op.

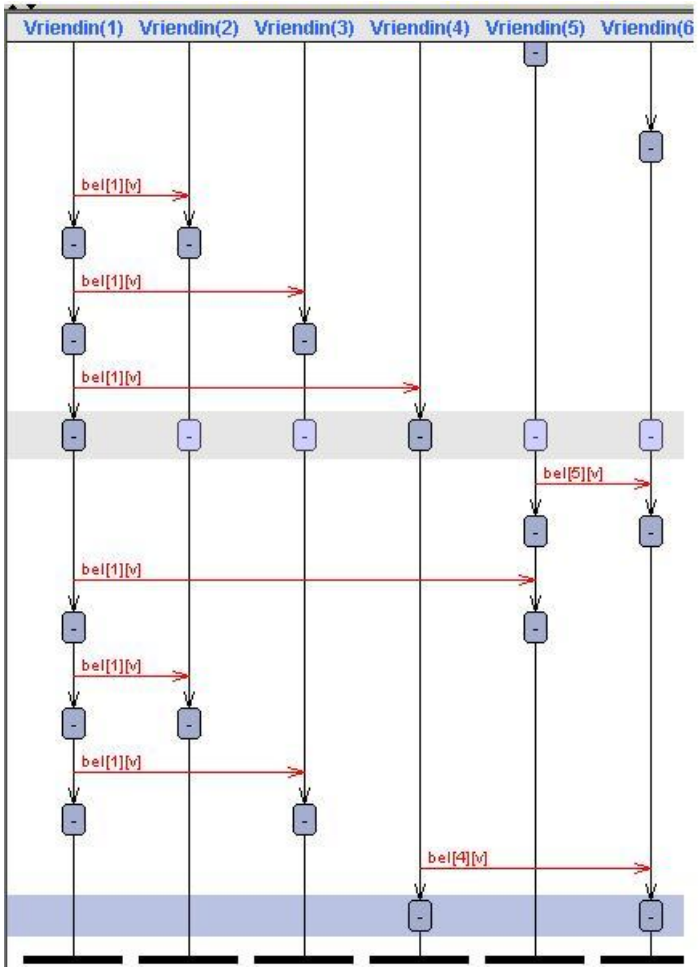
Wanneer twee vriendinnen elkaar spreken wisselen ze alle roddels uit die ze op dat moment weten.

Hoeveel gesprekken zijn nodig voordat iedereen alle roddels kent?

Toestandsdiagram



Oplossing Model Checker



Toepassingen van Model Checking

- Draadloze sensornetwerken
- NASA DEEP SPACE 1 missie
- Stormvloedkering bij Rotterdam
- Copieermachines
- Radarsysteem voor auto's
- ….