Model Checking for Autonomy Software

Charles Pecheur

RIACS / ASE Group, NASA Ames

Contents

Model Checking for Autonomy Software

- Why?

  Autonomy software, how to verify it?

- What?

  A bird's-eye view of model checking

- How?

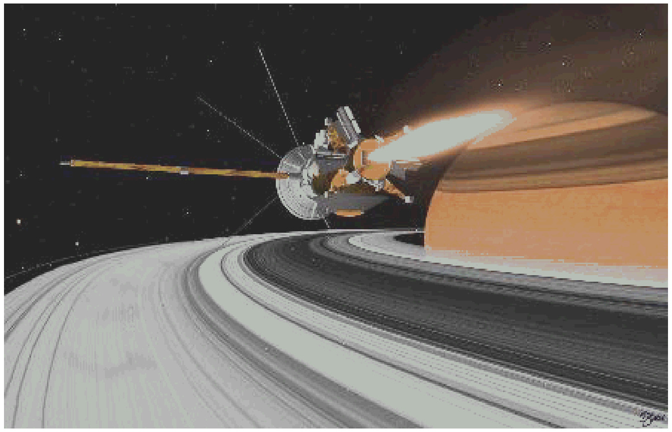
  Experiences in the ASE Group

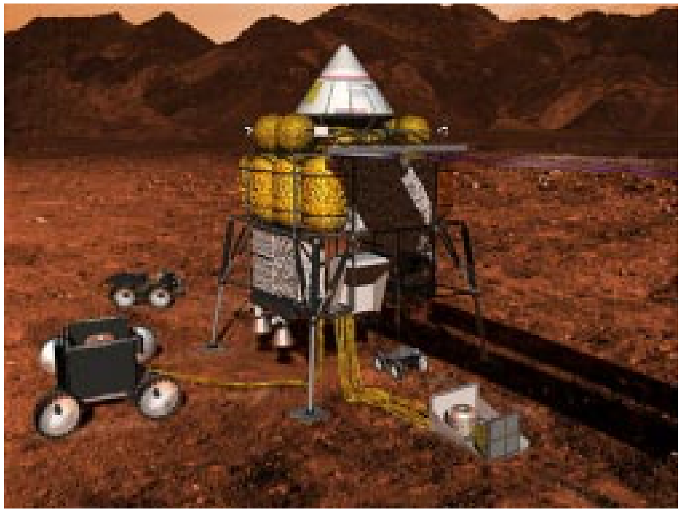
Autonomous Systems

"Faster, better, cheaper" spacecrafts

=> add on-board intelligence

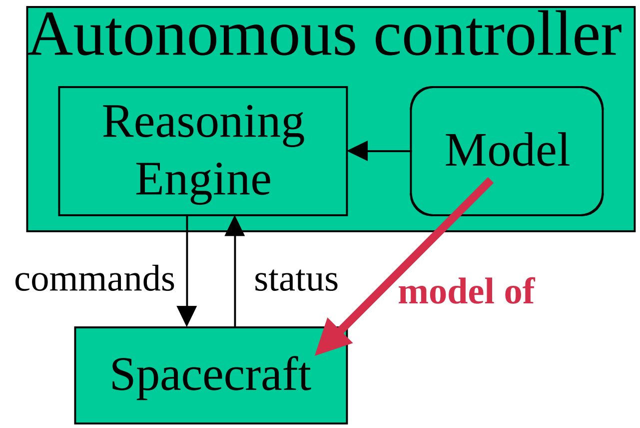
- From self-diagnosis
  to on-board science.
- Smaller mission control crews
  => reduced cost
- Less reliance on control link
  => OK for deep space





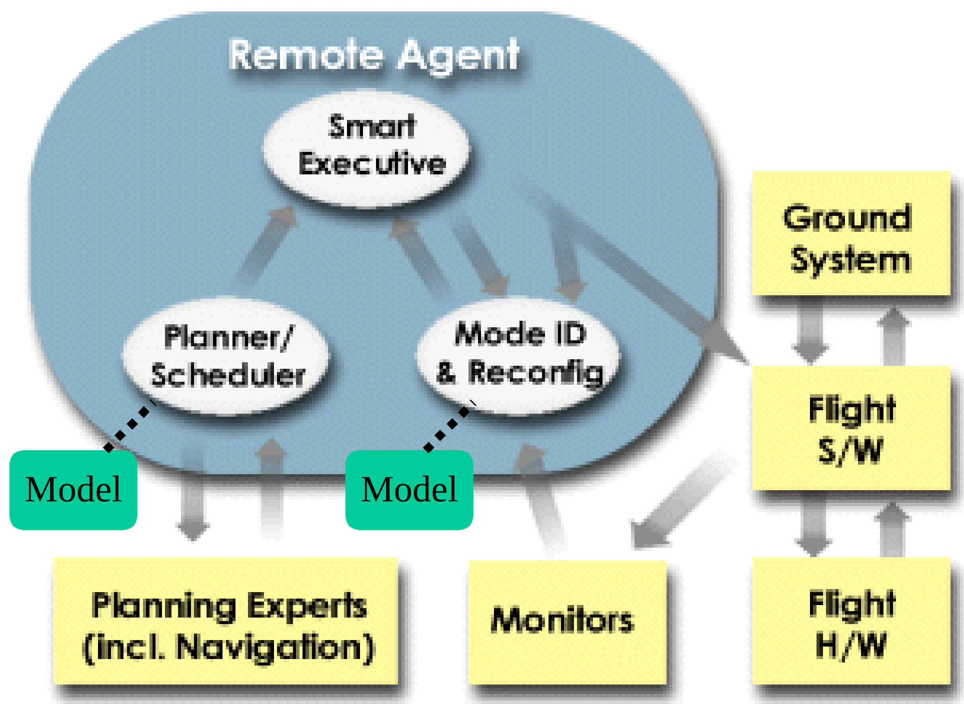
Model-Based Autonomy

- Based on AI technology
- General reasoning engine + application-specific model
- Use model to respond to unanticipated situations

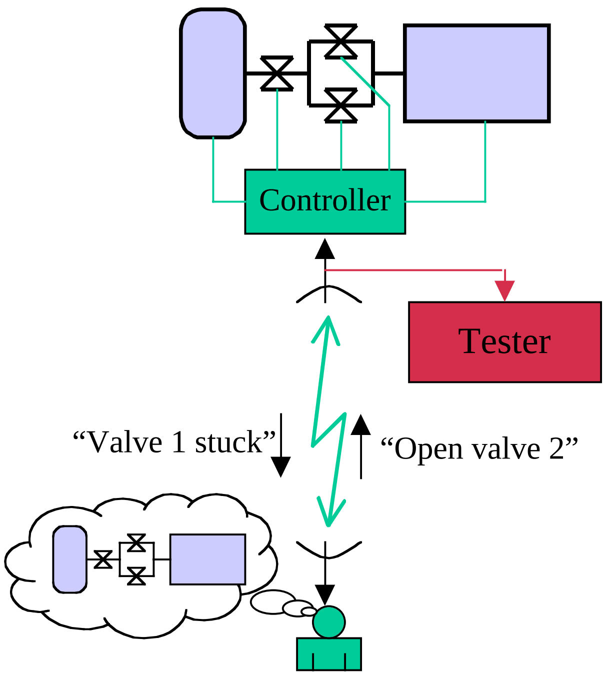


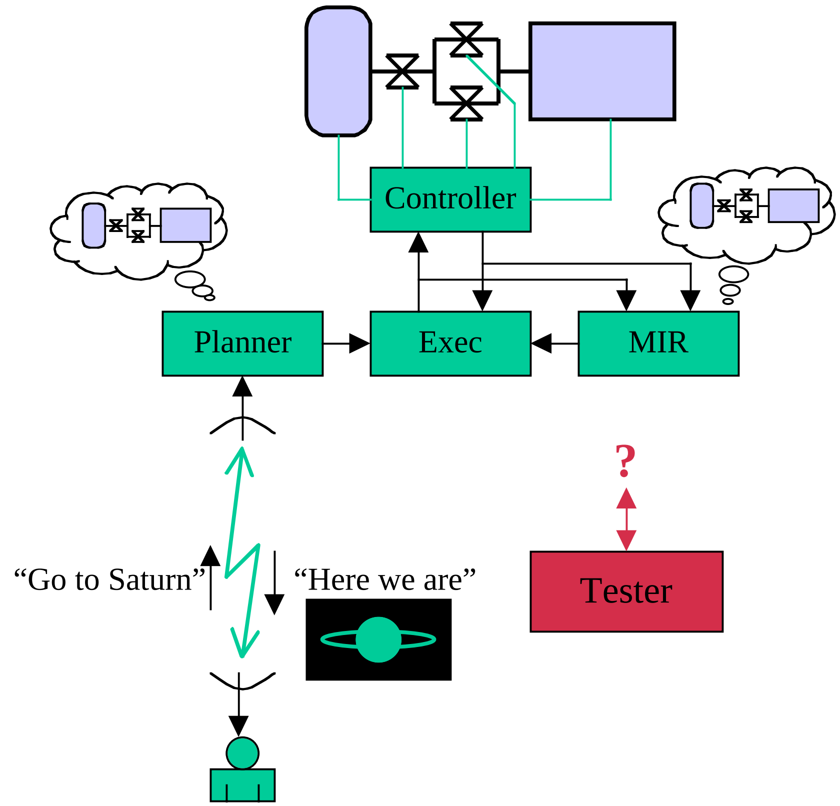
Example: Remote Agent

- From Ames ARA Group (+ JPL)
- On Deep Space One in May 1999 (1st AI in space!)



Controlled vs. Autonomous





Testing Autonomy Software?

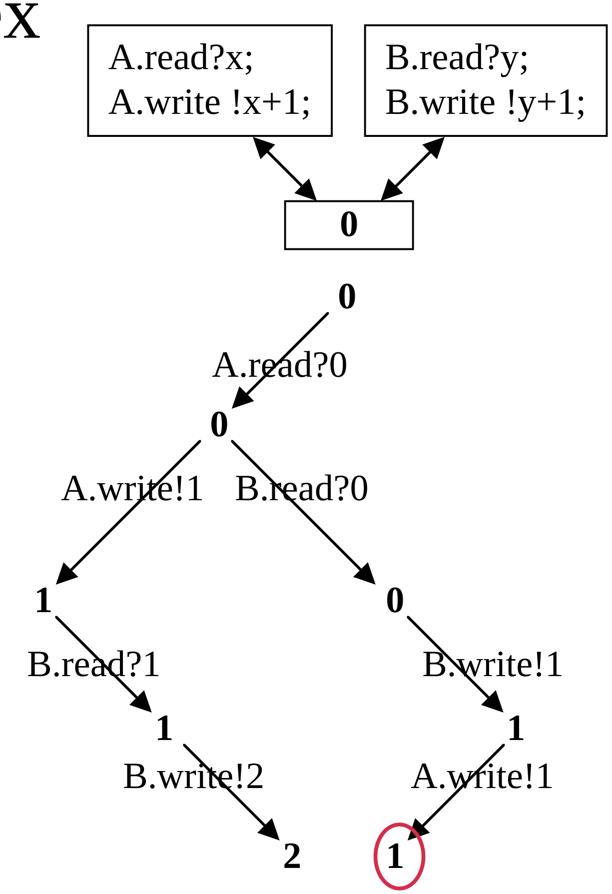
- Programs are much more complex
- Many more scenarios

  => testing gives low coverage

- Concurrency!

  Due to scheduling,
  the same inputs (test) can give
  different outputs (results)

  => test results are not reliable



Contents

Model Checking for Autonomy Software

- Why?

  Autonomy software, how to verify it?

- What?

  A bird's-eye view of model checking

- How?

  Experiences in the ASE Group

Model Checking

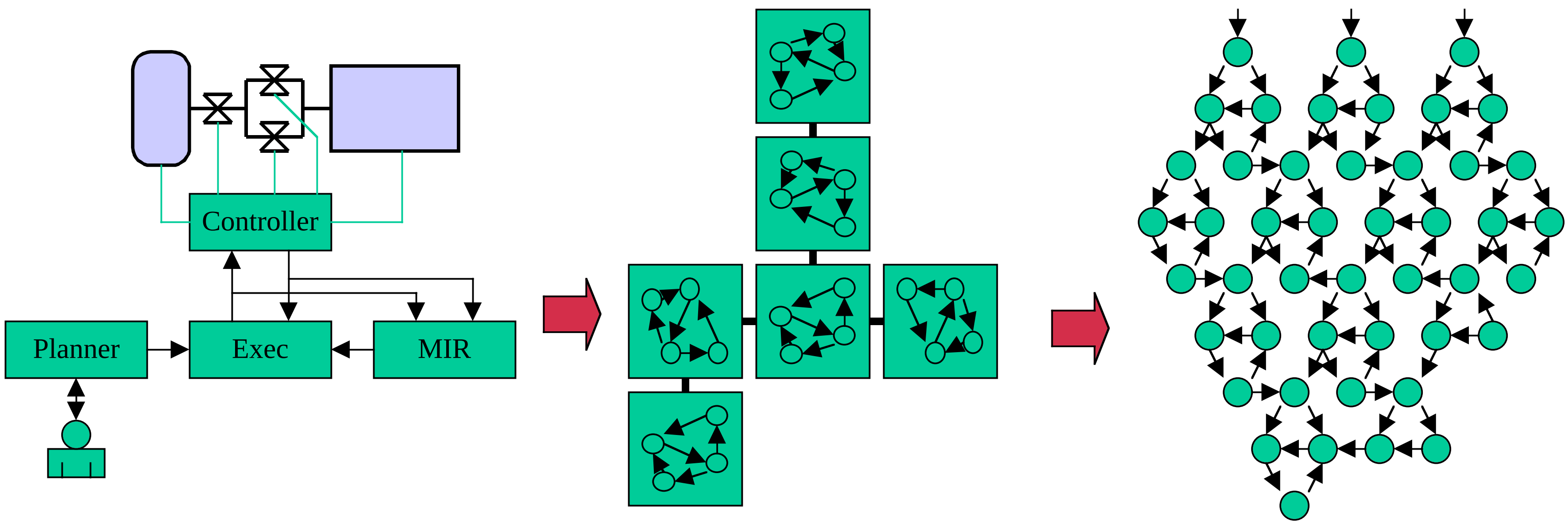
Check whether a system S satisfies a property P by exhaustive exploration of all executions of S

- Controls scheduling => better coverage
- Can be done at early stage => less costly
- Widely used in hardware, coming in software
- Examples: Spin (Bell Labs), Murphi (Stanford)

Model ...

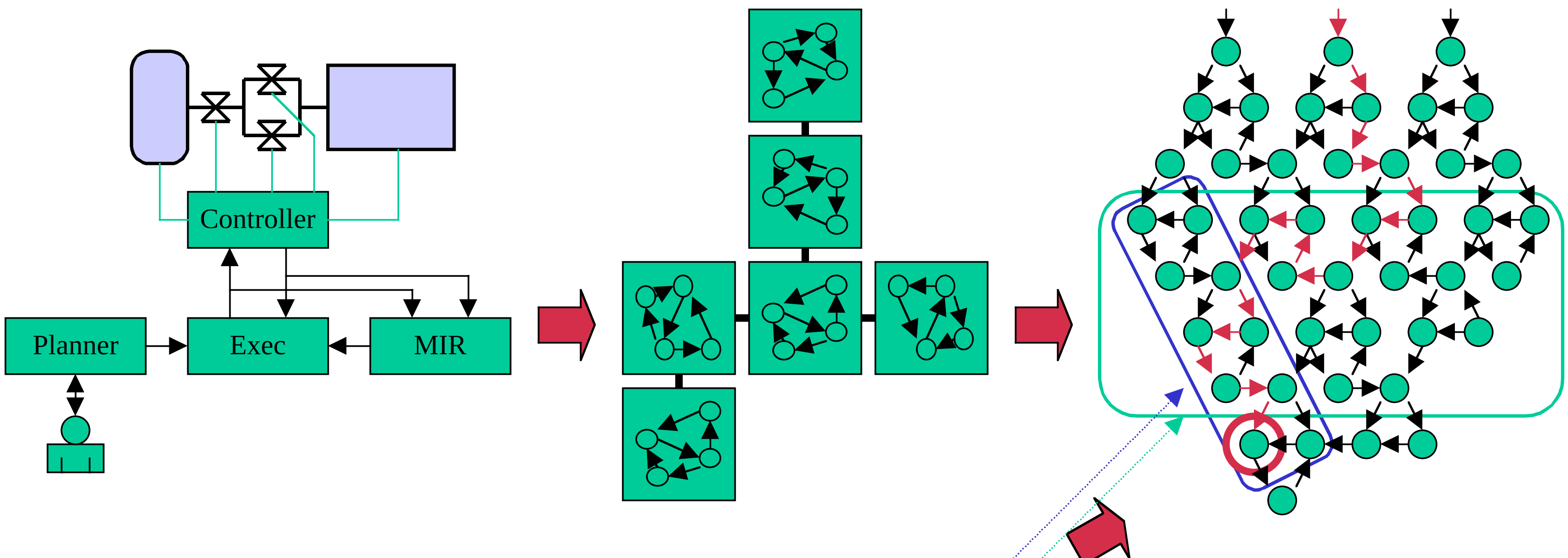
Modeling Abstraction

Verification



Model Checking

Modeling Abstraction **Verification**



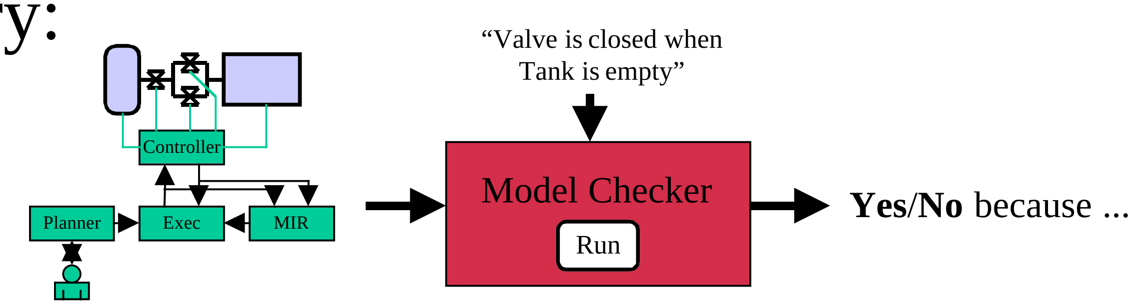
"Valve is closed when
Tank is empty"

AG (tank=empty
=> valve=closed)

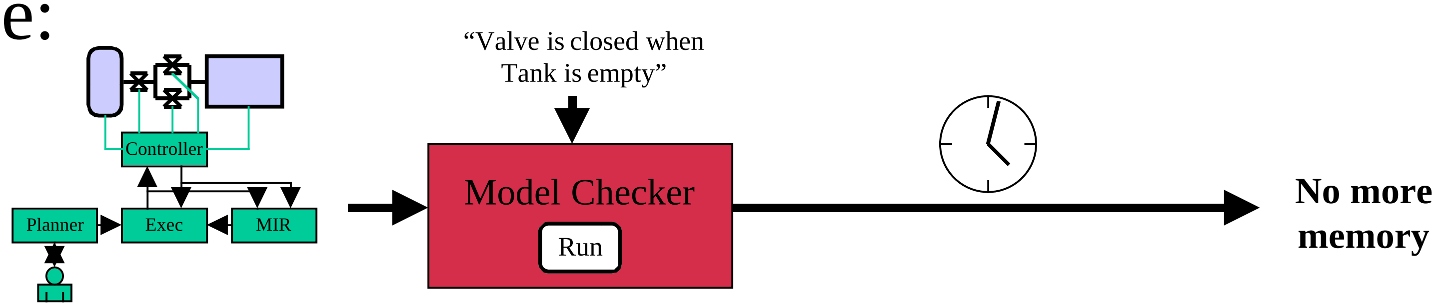
State Space Explosion

K processes with N local states $\leq$ N^K global states

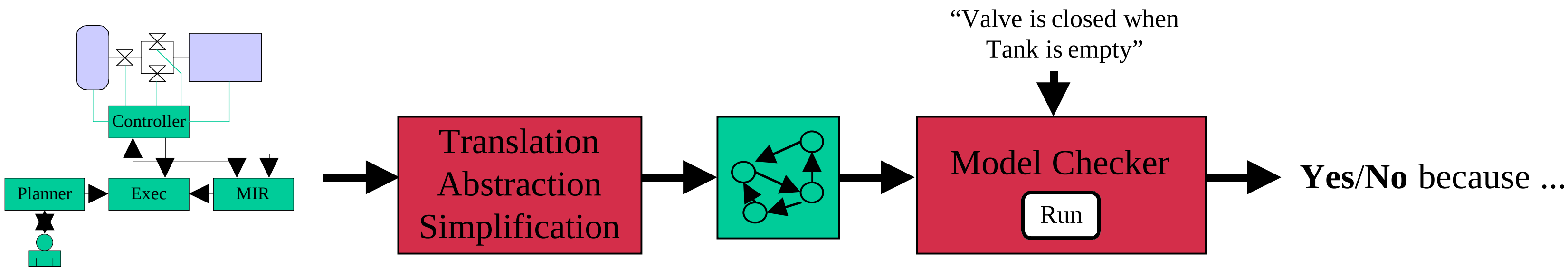
Theory:



Practice:



Modeling



This is the tough job!

- **Translation**: to model checker's syntax

  e.g. C —> Promela (Spin)

- **Abstraction**: ignore irrelevant parts

  e.g. contents of messages

- **Simplification**: downsize relevant parts

  e.g. number of processes, size of buffers

Temporal Logic

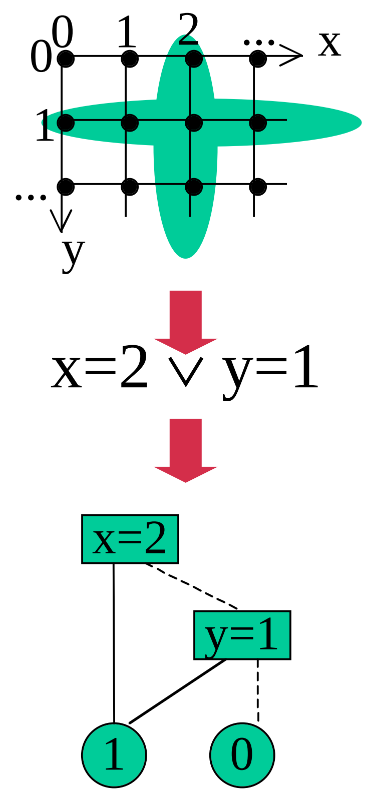
- Propositional logic + quantifiers over executions
- Example: "every request gets a response"

  AG (Req => **AF** Resp)

  Always Globally, if Req then **Always Finally** Resp
- Branching (CTL) vs. linear (LTL)
  - different verification techniques
  - neither is more general than the other
- Model checking without TL
  - Assertions, invariants
  - Compare systems, observers

Symbolic Model Checking

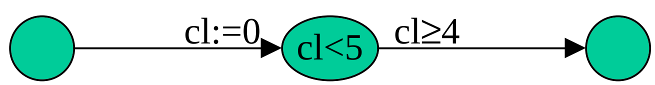
- Manipulates sets of states,
  Represented as boolean formulas,
  Encoded as binary decision diagrams.
- Can handle larger state spaces (10^{50} and up).
- BDD computations:
  - Good in average but exponential in worst case.
  - Computation time depends on BDD size
    => number of variables, complexity of formulas,
    but not directly state space size.
- Example: SMV (Carnegie Mellon U.)



Real-Time and Hybrid

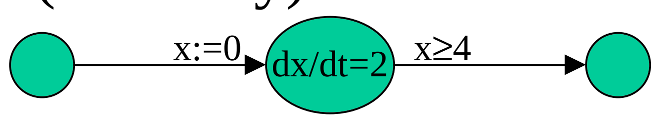
- "Classic" model checking: finite state, un-timed
- Real-time model checking: add clocks

  e.g. Khronos (Verimag), Uppaal (Uppsala/Aalborg)



- Hybrid model checking: add derivatives

  e.g. Hytech (Berkeley)



More complex problems & less mature tools

Contents

Model Checking for Autonomy Software

- Why?

  Autonomy software, how to verify it?

- What?

  A bird's-eye view of model checking

- How?

  Experiences in the ASE Group

Verification of Remote Agent Executive

(Lowry, Havelund and Penix)

- Smart executive system with AI features (Lisp)
- Modeled (1.5 month) and
  Model-checked with Spin (less than a week)
- 5 concurrency bugs found, that would have been hard to find through traditional testing

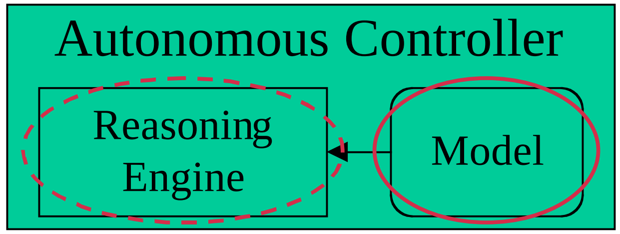
Hunting the RAX Bug

(Lowry, White, Havelund, Pecheur, ...)

- 18 May 1999: Remote Agent Experiment suspended following a deadlock in RA EXEC
  => Q: could V&V have found it?
- Over-the-week-end "clean room" experiment:
  - Front-end group selects suspect sections of the code
  - Back-end group does modeling (in Java) and verification (using Java Path Finder + Spin)
- => A: V&V found it... two years ago!

  Same as one of the 5 concurrency bugs found before
- Morale: Testing not enough for concurrency bugs!

Verification of Model-Based Autonomy



Reasoning Engine

- Relatively small, generic algorithm => use prover
- Requires V&V expert level but once and for all
- At application level, assume correctness (cf. compiler)

Model

- Complex assembly of interacting components => model checking
- Avoid V&V experts => automated translation
  Not too hard because models are abstract

Reasoning Engine + Model ???

Verification of Planner/Scheduler Models

(Penix, Pecheur and Havelund)

- Model-based planner from Remote Agent

  Models: constraint style, real-time

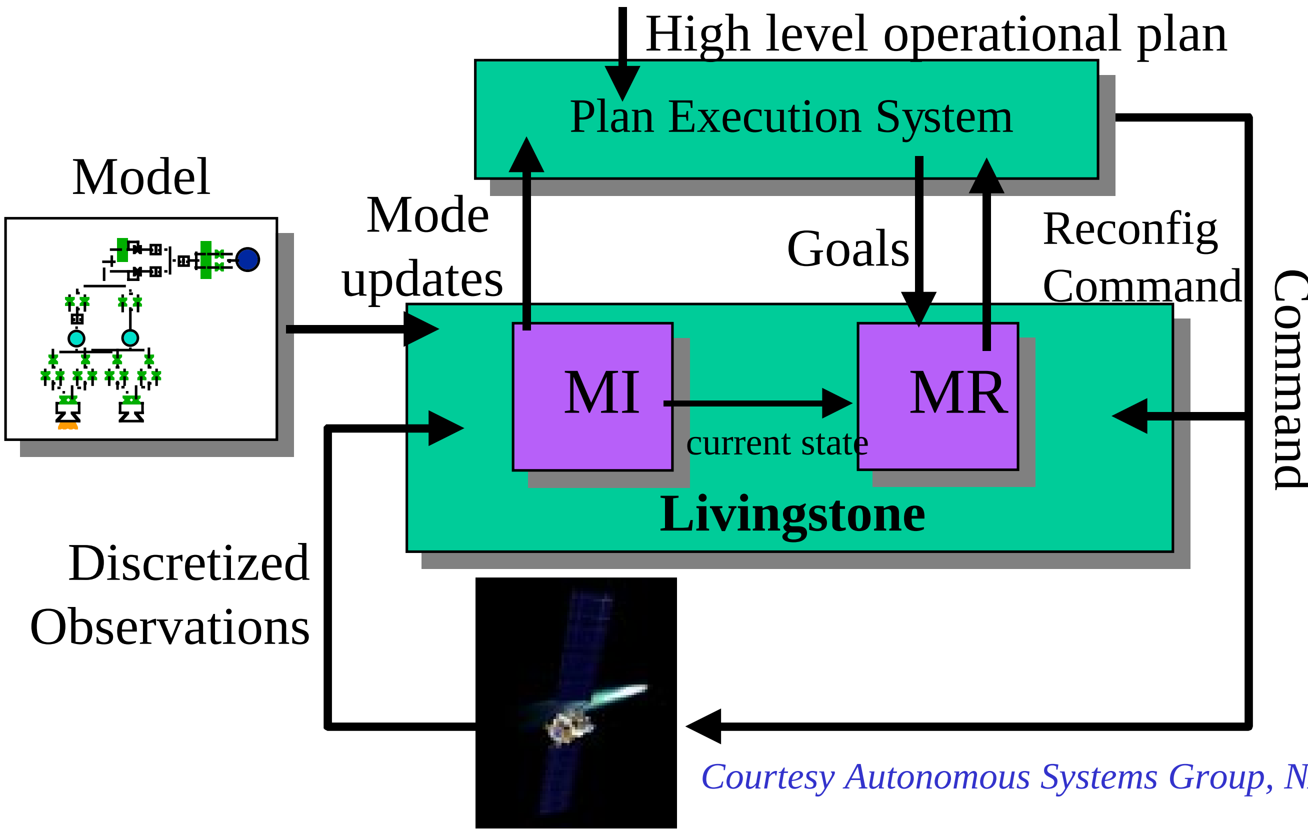
- Small sample model translated by hand

  Subset of the full modeling language, untimed

- Compare 3 model checkers: Spin, Murphi, SMV
  => SMV much easier and faster (≈0.05s vs. ≈30s)
- Continuation *(Khatib)*: handle timed properties using real-time model checker (Uppaal)

The Livingstone MIR

Remote Agent's model-based fault recovery sub-system



Courtesy Autonomous Systems Group, NASA Ames

Livingstone to SMV Translation

Livingstone Model

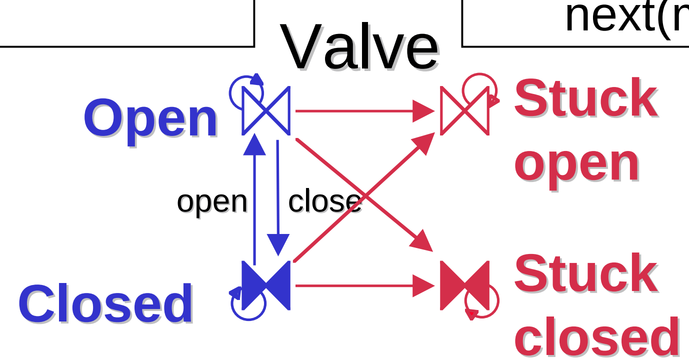
```
(defcomponent valve ()
  (:inputs (cmd :type valve-cmd))
 ...
  (Closed :type ok-mode
    :transitions
      ((do-open :when (open cmd)
        :next Open) ...))
  (StuckC :type :fault-mode ...)
  ...)
```

Livingstone Autonomous Controller

SMV Model

```
MODULE valve
VAR      mode: {Open,Closed,
                     StuckO,StuckC};
         cmd: {open,close};
DEFINE faults:={StuckO,StuckC};
TRANS
  (mode=Closed & cmd=open) ->
    (next(mode)=Open |
      next(mode) in faults)
```

SMV Symbolic Model Checker



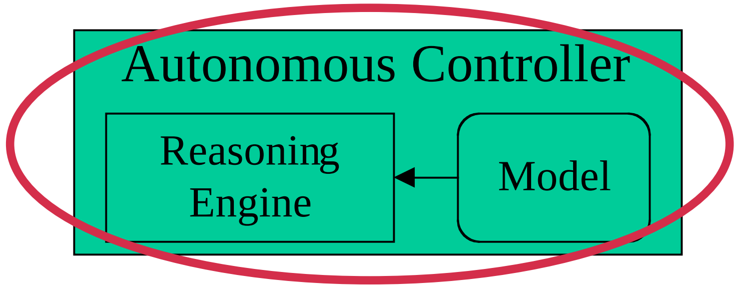
From Livingstone Models to SMV Models

(Simmons, Pecheur)

Translation program developed by CMU and Ames

- 4K lines of Lisp
- Similar nature => translation is easy
- Properties in temporal logic + pre-defined patterns
- Pilot Application:
  ISPP autonomous controller (KSC)
- In progress:
  - more property patterns
  - translate results back to Livingstone

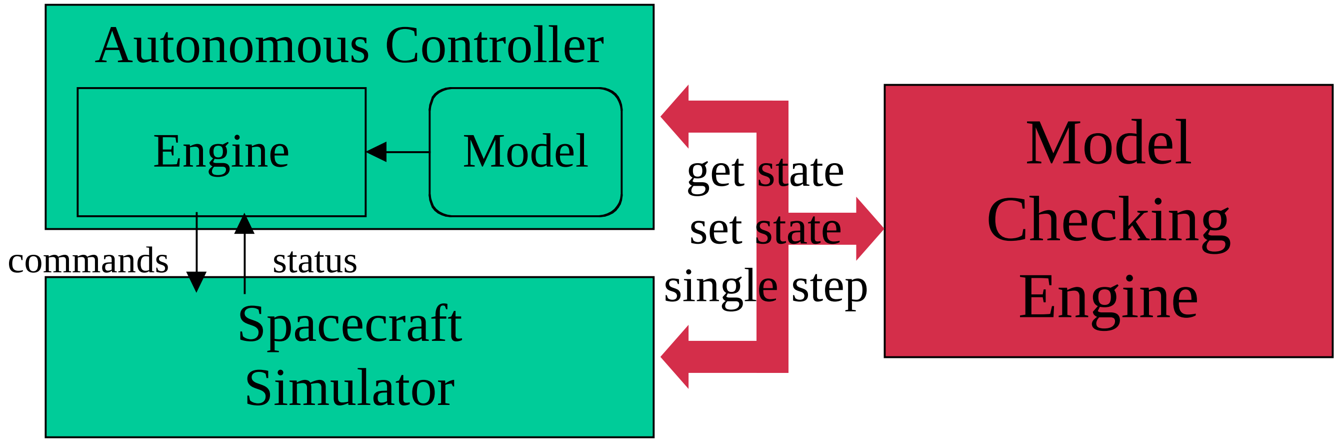
Verification of Model-Based Systems



- Model-based system = engine + model
- correct engine + correct plan ≠> good system !
  e.g. can fail to properly recognize a fault
- Model check? Very hard!

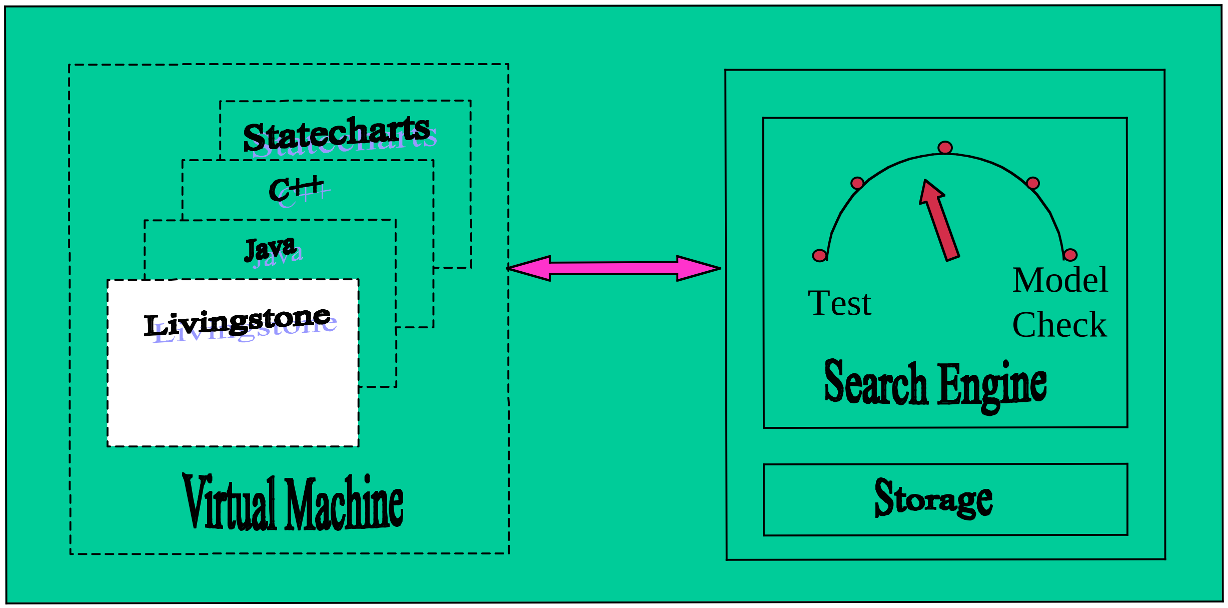
  Need (abstract) model of reasoning engine + model
  => complex, error-prone, huge state space

Analytic Testing



- Testing the real system => accuracy.
- Model-checking approach => exhaustive exploration.
- Restricted scenarios in simulator (otherwise too big).
- Completes, not supersedes, Model V&V (later stage).

Generic Verification Environment



- Principle: uncouple V&V subject from V&V algo.
- Common denominator of several projects in ASE.
- Hooks already present in Livingstone.

Conclusions

Model checking:

- Autonomy needs it – testing is not enough
- General pros&cons apply:
  - exhaustive... if model is small enough
  - automatic verification... but tough modeling
- Works nicely on autonomy models
- Solutions inbetween testing and model checking
- Not short of tough problems:
  - Real-time, hybrid, AI
  - Learning/adaptive systems: *after* training/*including* training

MPL to SMV: Example

Lisp shell

```
(load "mpl2smv.lisp")
;; load the translator
;; Livingstone not needed!

(translate "ispp.lisp" "ispp.smv")
;; do the translation

(smv "ispp.smv")
;; call SMV
;; (as a sub-process)
```

ispp.lisp

```
(defcomponent  heater …)
(defmodule valve-mod …)
…
(defverify
  :structure (ispp)
  :specification (all (globally …)))
```

ispp.smv

```
MODULE Mheater …
MODULE Mvalve-mod …
…
MODULE main
VAR Xispp:Mispp
SPEC AG …
```

SMV output

```
Specification AG … is false as shown …
State 1.1: …
State 1.2: …
```