Reminder: MoVES Doctoral Lecture

Applications of Computer Science to Space Applications
Tuesday June 12, 2007
Institut d’Informatique (Séminaire 3), FUNDP, Namur

- Software Verification for Space Applications
  Dr. Guillaume Brat, RIACS / NASA Ames, USA

- Planning and Scheduling for Space Applications
  Ir. Marc Niézette, Vega Group PLC, Germany

Please register by e-mail to Charles.Pecheur@uclouvain.be
http://prog.vub.ac.be/moves/info/courses
Verification of Embedded Control Software
MoVES FUNDP-UCL Presentation Meeting
25 May 2007

Charles Pecheur
Dept. INGI, UC Louvain
charles.pecheur@uclouvain.be
http://www.info.ucl.ac.be/~pecheur/
Embedded Controllers

- Everywhere
  - more and more so
- Dependability is critical
  - human risks
  - material risks
  - economic risks
- Logic (vs. physical) part is increasing
Process Control

- Partially observable process (hidden state $x$, estimated by $\hat{x}$)

- **observability**: infer $x$ from $y$ (and $u$)
- **commandability**: impose $x$ through $u$

- **control theory**: $x =$ physical quantities, differentiable → linear models, PDI controllers
- **logic processes**: $x =$ states, modes, **failures**, discrete → state machines, programmable automata
Process Models

Useful for:

- Safety analysis
- Model-based control
  - planning/scheduling
  - diagnosis/recovery
- Controller verification
- Controller synthesis

Features:

- Strongly concurrent, loose coupling (faults!), non-deterministic
- Time-dependent, continuous/discrete, stochastic

State space explosion at its worst
Symbolic techniques are a must
Symbolic Model Checking

• **Symbolic** model checking =
  – compute sets of states,
  – using symbolic representations,
  – that can be efficiently encoded and computed.

• Can handle very large state spaces \((10^{50+})\), or even infinite domains (continuous time and variables)

• Example: **SMV/NuSMV** (Carnegie Mellon/IRST)
  – finite state using boolean encoding (BDD, SAT)
Livingstone-to-SMV Translator

Translator hides away SMV, offers a model checker for Livingstone

Working prototype (Java), applied to several real-size applications (chemical process, propulsion, aerospace)

Extended for verification of diagnosability

GUI, integration with other Livingstone development tools

Generic concept, applicable to other languages and tools
**Diagnosability**

- **Diagnosis**: estimate the hidden state $x$ (incl. failures) given observable commands $u$ and sensors $y$.
- **Diagnosability**: Can (a smart enough) Diagnoser always tell when Process comes to a bad state?
- **Property of the Process** (not the Diagnoser)
  - even for non-model-based diagnosers
  - but analysis needs a (process) model
Verification of Diagnosability

joint work with Alessandro Cimatti

- **Intuition**: bad is diagnosable if and only if there is no pair of trajectories, one reaching a bad state, the other reaching a good state, with identical observations.
  - or some generalization of that: (context, two different faults, ...)

- **Principle**:
  - consider two concurrent copies \( x_1, x_2 \) of the process, with coupled inputs \( u \) and outputs \( y \)
  - check for reachability of \((\text{good}(x_1) \&\& \text{bad}(x_2))\)

- Back to a classical (symbolic) model checking problem!
- Supported by Livingstone-to-SMV translator
Epistemic Logic

- Reasoning about knowledge
  \[ K_a \varphi = \text{agent } a \text{ knows } \varphi \]
- Interpreted over an *Interpreted System*
  - *Transition system* \( T \)
  - *Observation functions* \( \text{obs}_a(\sigma) \) over runs \( \sigma \) of \( T \)
  - \( K_a \varphi \) holds after \( \sigma \) iff \( \varphi \) holds after all \( \sigma' \) such that \( \text{obs}_a(\sigma) = \text{obs}_a(\sigma') \)
- **CTLK = temporal + epistemic logic**
- **Observational view**: agents observe current state \( s \) only
  - \( K_a \varphi \) holds in \( s \) iff \( \varphi \) holds in all \( s' \) such that \( \text{obs}_a(s) = \text{obs}_a(s') \)
  - symbolic model checking can be generalized from CTL to CTLK

© Ch. Pecheur, UC Louvain, May 2007
Diagnosability and CTLK

Joint work with Franco Raimondi

Considering the diagnoser as an agent $D$ observing the system,

Fault $F$ is diagnosable

iff

$$AG (K_D F \lor K_D \neg F)$$

- CMAS: symbolic model checker for CTLK
  - developed by Franco Raimondi
  - very crude modeling language

- CTLK in NuSMV
  - leverages SMV's rich modeling language
  - via reduction to CTL + actions
  - see [Raimondi-Pecheur-Lomuscio 2005]

- Limitation: "memory-less" diagnosis (because observational view)
NuSMV with Actions

• **CTLK** model checking on a transition system reduces to **CTL** model checking on a **labeled** transition system
  – ≈ action-based ≈ event-based ≈ I/O automata: \( s \xrightarrow{a} s' \)
  – temporal transitions \( \tau \)
    + epistemic equivalences \( \sim_a \) for each agent
  – under the observational view

• NuSMV partially supports labeled transitions
  – allowed in models \( \textbf{IVAR} \) but not in (CTL) properties

• We have added action-based (branching) logics to NuSMV
  – \( \textbf{ARCTL} : A_{\alpha} G \varphi = \) For all \( \alpha \)-paths, \( \varphi \) always hold
  – Others easily added, e.g. ACTL [deNicola-Vaandrager]

• To do: fairness on finite paths, counter-example generation
Perspectives

• Tools and case studies
  – scaling up, fitness for purpose
  – other fields: man-machine interfaces, security protocols, ...

• Finer / richer models
  – continuous, real-time, hybrid, stochastic
  – needs SAT solving, QBF, decision procedures, CSP, ...

• Risk analysis
  – Use / verify / generate fault trees, FMEA, ...

• Extended logics
  – Distributed diagnosis = common/joint knowledge
  – Fault recovery / general control = game theory
  – A lot of existing results but few extended tools
Summary

• Capability:
  
  **Analysis of control systems using model checking**

• Aspects:
  – observability/commandability
  – failure diagnosis and recovery
  – discrete / continuous / timed / hybrid models
  – epistemic logic, action-based temporal logic

• The Big Picture:
  
  Apply and extend **model checking**
  to **observability-related** concerns

• To learn more: