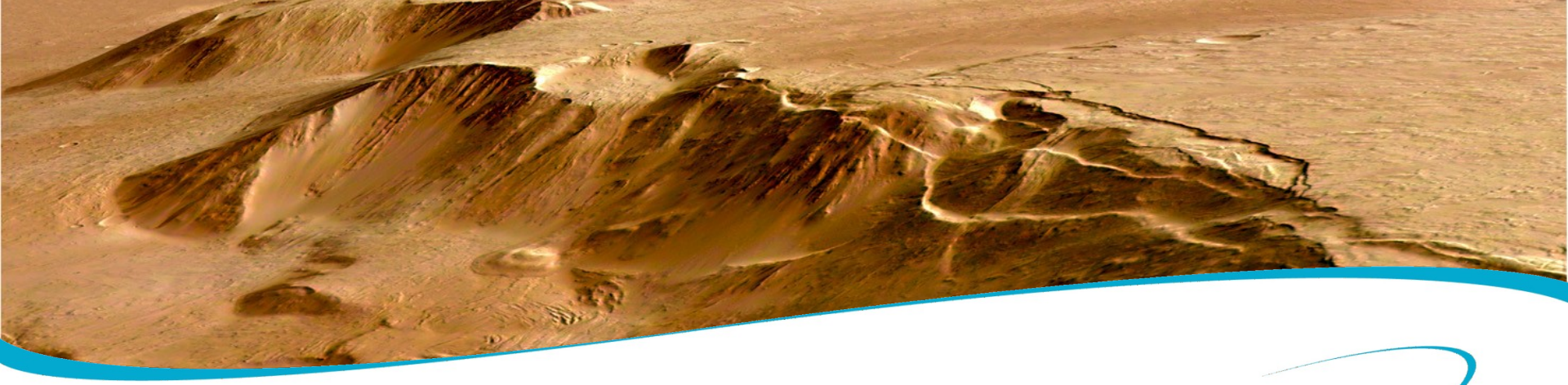


OMC-ARE
On Board Model Checking
Autonomous Reasoning Engine

ESA-ESTEC Noordwijk , 10/11/2008

OMC-ARE

On-board Model Checking Autonomous Reasoning Engine

Alessandro Cimatti[1] Marco Roveri[1] Andrea Guiotto[2]

[1] Fondazione Bruno Kessler (Italy)

[2] Thales Alenia Spazio (Italy, France)

ESA/ESTEC ITT AO/1-5184/06/NL/JD - ON BOARD MODEL CHECKING

http://es.fbk.eu/projects/esa_omc-are

Outline

- Introduction
- Project objectives
- The proposed solution: ARE
  - The ARE architecture
  - The ARE formal framework
- Evaluation approach
- Related work
- Industrial perspective
- Conclusion
- Future work
- Demo
- Discussion

Introduction

Andrea Guiotto

The consortium at a glance

Thales Alenia Space Italia
Prime contractor

- Mission scenarios and requirements
- Architecture design
- Validation
- Evaluation & characterization
- Project management

Fondazione Bruno Kessler
Subcontractor

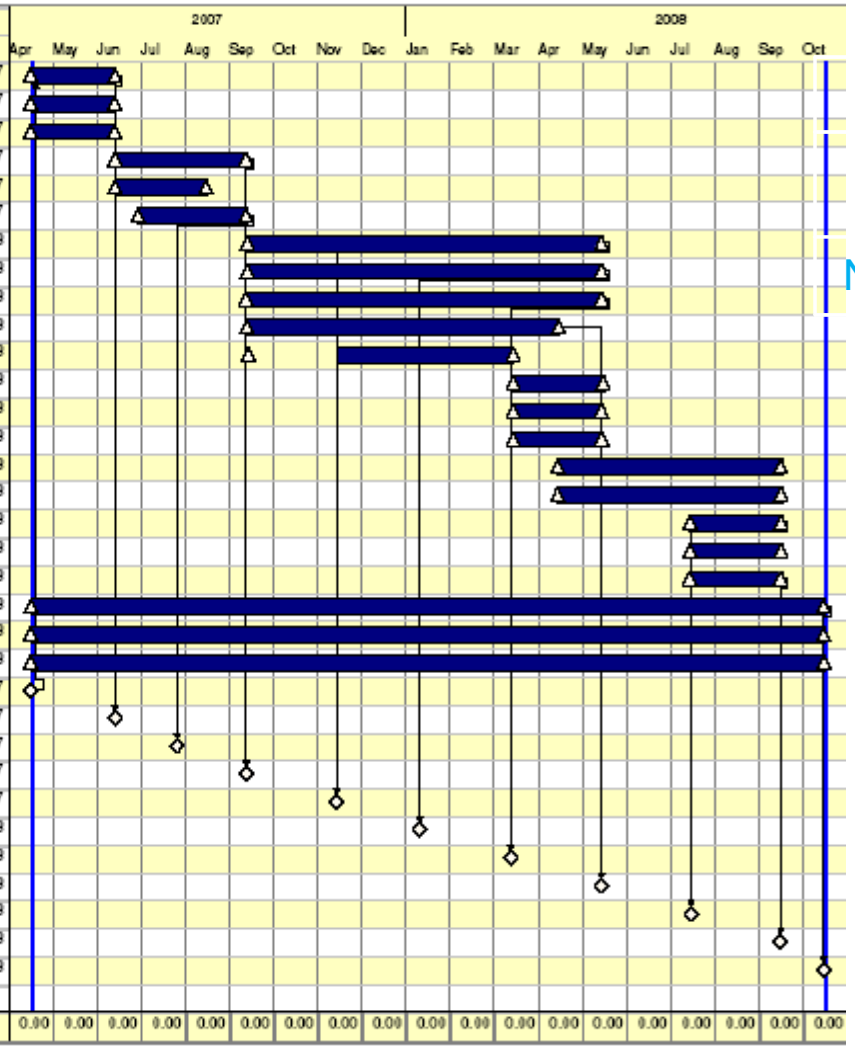
- Technology survey
- Formal Framework definition
- Architecture design
- Algorithms definition
- Implementation of the ARE
- Development of formal models
- Support to evaluation & characterization

Thales Alenia Space France
Subcontractor

- S/C Simulator

Initial schedule

ID	Name	Start	Finish
1000	Production of Req. Baseline	16/04/2007	14/06/2007
1100	Autonomy Needs & Solution	16/04/2007	14/06/2007
1200	Requirement Baseline	16/04/2007	14/06/2007
2000	Production of Technical Specification	14/06/2007	13/09/2007
2100	SW Specification	14/06/2007	16/08/2007
2200	UML Architecture	30/06/2007	13/09/2007
3000	Implementation of the ARE	14/09/2007	15/05/2009
3100	SW Development	14/09/2007	15/05/2009
3110	On-board infrastruct., I/F & SW integratio	13/09/2007	15/05/2009
3120	ARE Development	14/09/2007	15/04/2009
3130	Spacecraft Simulator Configuration	15/11/2007	14/03/2009
3200	SW Validation	15/03/2009	16/05/2009
3210	System SW Validation	15/03/2009	15/05/2009
3220	Support to SW Validation	15/03/2009	15/05/2009
4000	Performance Evaluation	15/04/2009	16/09/2009
4100	Evaluation of Case Studies	15/04/2009	16/09/2009
4200	Characterization of the approach	15/07/2009	16/09/2009
4210	System Characterization	15/07/2009	16/09/2009
4220	Technology Characterization	15/07/2009	16/09/2009
5000	Management and Reporting	16/04/2007	16/10/2009
5100	AAS Management	16/04/2007	16/10/2009
5200	FBK Management	16/04/2007	16/10/2009
	TO	16/04/2007	16/04/2007
	SRR	14/06/2007	14/06/2007
	PM1	27/07/2007	27/07/2007
	PDR	13/09/2007	13/09/2007
	PM2	15/11/2007	15/11/2007
	PM3	11/01/2009	11/01/2009
	PM4	13/03/2009	13/03/2009
	CDR	15/05/2009	15/05/2009
	PM5	16/07/2009	16/07/2009
	AR	16/09/2009	16/09/2009
	FP	16/10/2009	16/10/2009



KO: 16/04/2007

FP: 16/10/2008

New FP: 10/11/2008

Motivations

- **Limits of the traditional approaches**:
  1. The control of a spacecraft takes place mostly from ground, through the exchange of sequences of low level commands.
  2. Spacecraft is typically unable to deal alone with unexpected events from the environment or unpredicted on-board failures.
  3. In deep space and remote planetary exploration missions the limits in communication between ground and spacecraft (in time and bandwidth) increase reaction times and can decrease the efficiency of corrective actions.

Motivations (II)

- **Enhancements with ARE:**
  1. Providing remote systems with the ability to create their own plans based on up-to-date information and enabling them to re-plan in response to dynamic events would greatly improve the efficiency of a mission and potentially improve the safety of systems.
  2. Ground operators can use the restricted communication link to forward high-level mission objectives, which the on-board system can turn into detailed commands.
  3. Execution can be monitored continuously and re-planning invoked when any execution problem occurred.

Study objectives

- Demonstrate the applicability of
  - model based reasoning, and
  - model checking techniques
- … to increase autonomy of on-board reasoning
  - on-board re-planning
  - on-board plan validation
  - monitoring
  - FDIR

Study objectives (II)

- Find out and justify the place of model checking in space systems to increase
  - degree of autonomy
  - confidence on the correctness
- Develop a SW prototype, named Autonomous Reasoning Engine (ARE)
  - to be used as building block in future space missions
- Evaluate the global approach on real case studies
  - Planetary rover
  - Orbiter spacecraft
- Characterize the approach w.r.t. peculiarities of the space environment.
  - Memory usage, speed, embeddability in space architectures

FONDAZIONE
BRUNO KESSLER

ThalesAlenia
Space

From Model Checking to Model Based Reasoning for Autonomy

Alessandro Cimatti

FBK-irst

- Fondazione Bruno Kessler
  - private research foundation with public finalities
  - formerly Istituto Trentino di Cultura
  - funded by the Autonomous Province of Trento
  - several research centers
- Center for Information Technologies
  - about 200 researchers in advanced areas of IT
  - vision, acoustics, automated speech and translation, software engineering, knowledge management
- The Embedded Systems Unit
  - about 20 people
  - research staff, programmers, ph.d students

Embedded Systems Unit

- Activities
  - research
  - tools
  - technology transfer
- Three main lines of research
  - formal specification and verification
  - reasoning for autonomy
  - distributed wireless sensor networks
- Some partnerships in technology transfer / research projects
  - European Space Agency
  - European Railway Agency
  - Invensys Climate Controls
  - Intel
  - Ansaldo Trasporti
  - Ansaldo Segnalamento Ferroviario
  - Thales/Alenia Spazio

"Standard" Formal Verification

- Embedded systems carry out critical functions
  - lives and huge amounts of money at stake
- Bugs are possible
  - Radiotherapy Machine Therac 25: 6 people overradiated
  - Intel Pentium FDVI: 400 MUSD
  - ….

Critical Systems

- Research: Formal Verification, Model Checking
  - Advanced methods for design and development
- Find the greatest number of defects
- Certify the absence of defects

- The basic intuition
  - The system under analysis is modeled as a mathematical theory
  - Its correctness reduced to proving a theorem
  - Model checker: a specialized software tool that is able to
    - » prove a theorem
    - » show that a conjecture is not a theorem
  - automatic, esaustive, useful to the designer

- Formal verification
  - off line, design time technologies
  - avionics, space, railways, hardware
  - Functional verification, Requirements engineering

"Standard" Formal Verification

- Embedded systems carry out critical functions
  - lives and huge amounts of money at stake
- Bugs are possible
  - Radiotherapy Machine Therac 25: 6 people overradiated
  - Intel Pentium FDVI: 400 MUSD
  - ….

- Research: Formal Verification, Model Checking
  - Advanced methods for design and development
- Find the greatest number of defects
- Certify the absence of defects

- The basic intuition
  - The system under analysis is modeled as a mathematical theory
  - Its correctness reduced to proving a theorem
  - Model checker: a specialized software tool that is able to
    - » prove a theorem
    - » show that a conjecture is not a theorem
  - automatic, esaustive, useful to the designer

- Formal verification
  - off line, design time technologies
  - avionics, space, railways, hardware
  - Functional verification, Requirements engineering

"Standard" Formal Verification

- Embedded systems carry out critical functions
  - lives and huge amounts of money at stake
- Bugs are possible
  - Radiotherapy Machine Therac 25: 6 people overradiated
  - Intel Pentium FDVI: 400 MUSD
  - ….

- Research: Formal Verification, Model Checking
  - Advanced methods for design and development
- Find the greatest number of defects
- Certify the absence of defects

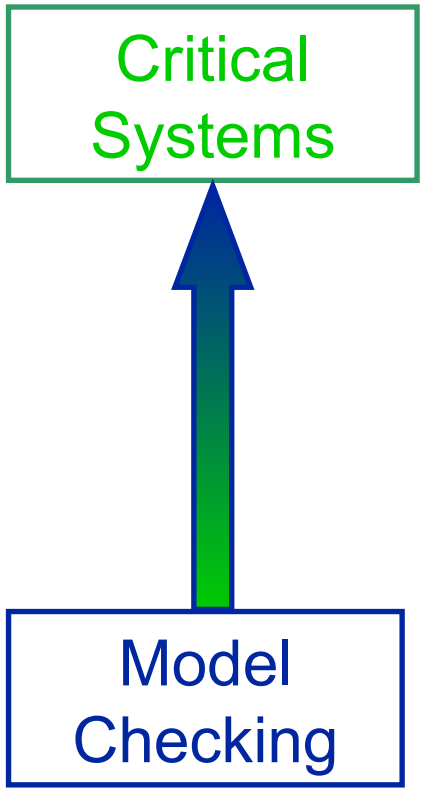
- The basic intuition
  - The system under analysis is modeled as a mathematical theory
  - Its correctness reduced to proving a theorem
  - Model checker: a specialized software tool that is able to
    - » prove a theorem
    - » show that a conjecture is not a theorem
  - automatic, esaustive, useful to the designer

Model Checking

- Formal verification
  - off line, design time technologies
  - avionics, space, railways, hardware
  - Functional verification, Requirements engineering

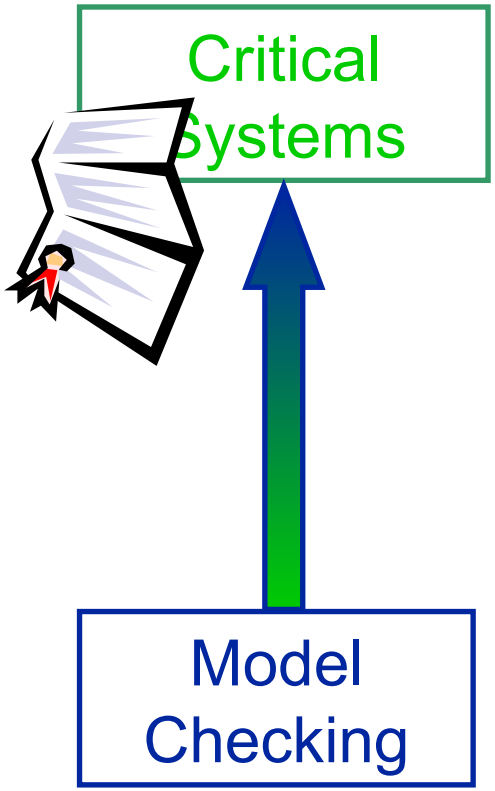
"Standard" Formal Verification

- Embedded systems carry out critical functions
  - lives and huge amounts of money at stake
- Bugs are possible
  - Radiotherapy Machine Therac 25: 6 people overradiated
  - Intel Pentium FDVI: 400 MUSD
  - ….
- Research: Formal Verification, Model Checking
  - Advanced methods for design and development
- Find the greatest number of defects
- Certify the absence of defects
- The basic intuition
  - The system under analysis is modeled as a mathematical theory
  - Its correctness reduced to proving a theorem
  - Model checker: a specialized software tool that is able to
    - » prove a theorem
    - » show that a conjecture is not a theorem
  - automatic, esaustive, useful to the designer
- Formal verification
  - off line, design time technologies
  - avionics, space, railways, hardware
  - Functional verification, Requirements engineering



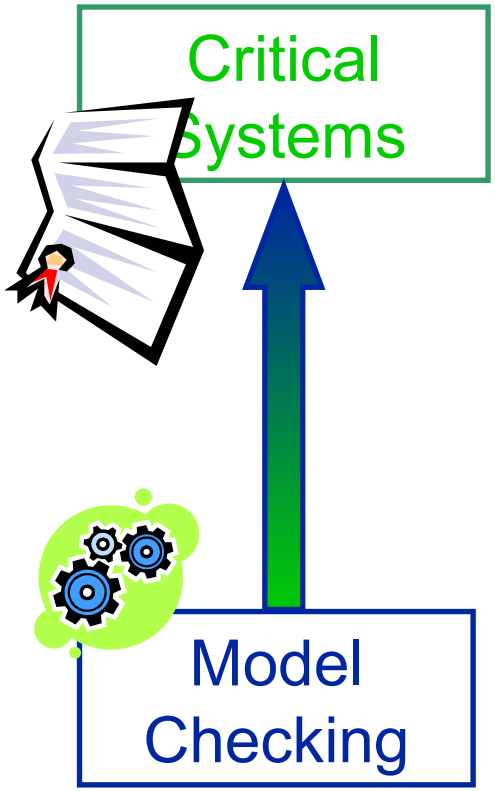
"Standard" Formal Verification

- Embedded systems carry out critical functions
  - lives and huge amounts of money at stake
- Bugs are possible
  - Radiotherapy Machine Therac 25: 6 people overradiated
  - Intel Pentium FDVI: 400 MUSD
  - ….
- Research: Formal Verification, Model Checking
  - Advanced methods for design and development
- Find the greatest number of defects
- Certify the absence of defects
- The basic intuition
  - The system under analysis is modeled as a mathematical theory
  - Its correctness reduced to proving a theorem
  - Model checker: a specialized software tool that is able to
    - » prove a theorem
    - » show that a conjecture is not a theorem
  - automatic, esaustive, useful to the designer
- Formal verification
  - off line, design time technologies
  - avionics, space, railways, hardware
  - Functional verification, Requirements engineering



"Standard" Formal Verification

- Embedded systems carry out critical functions
  - lives and huge amounts of money at stake
- Bugs are possible
  - Radiotherapy Machine Therac 25: 6 people overradiated
  - Intel Pentium FDVI: 400 MUSD
  - ….
- Research: Formal Verification, Model Checking
  - Advanced methods for design and development
- Find the greatest number of defects
- Certify the absence of defects
- The basic intuition
  - The system under analysis is modeled as a mathematical theory
  - Its correctness reduced to proving a theorem
  - Model checker: a specialized software tool that is able to
    - » prove a theorem
    - » show that a conjecture is not a theorem
  - automatic, esaustive, useful to the designer
- Formal verification
  - off line, design time technologies
  - avionics, space, railways, hardware
  - Functional verification, Requirements engineering



Results in Formal Verification

- Research line in Bounded Model Checking
  - In CiteSeer, *"20th Most cited source document published in 1999 as of September 2006"*
- The NuSMV model checker
  - more than 20000 downloads since 2004
  - Development based on OpenSource
  - Industrial users include Rockwell-Collins, Intel, NASA, Boeing
- The FSAP safety analysis system
  - fault tree analysis specialized in avionics
  - commercial version under industrial evaluation
- Recent research projects
  - ESACS (FP V), ISAAC (FP VI), MISSA (FP VII): avionics
  - PROSYD (FP VI): requirements analysis for hardware
  - ORCHID (PAT competitive funds): new generation verification tools
  - BOWLING: grant from the Academic Research Program by INTEL Corp.
  - COCONUT (FP VII): correctness by construction of embedded systems
  - COMPASS: competitive invitation to tender by ESA

Model-Based Autonomy

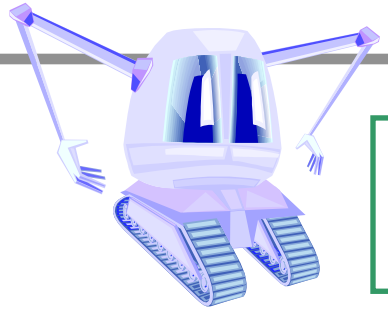
- Autonomous systems
  - Example: remote space systems
- How can we program such systems?
  - Potential for on-board autonomy
  - Required Functions
    - » Action planning
    - » Execution Monitoring
    - » Fault diagnosis
- Model-based Reasoning via Model Checking
  - Environment and functions to carry out modeled as mathematical theories
  - Functions for autonomy (planning, monitoring and diagnosis) carried out as proof search (model checking)
  - Extensions of verification algorithms
- From off-line to on-line
  - can be useful for ground operation
  - carried out on board

Model-Based Autonomy

- Autonomous systems
  - Example: remote space systems
- How can we program such systems?
  - Potential for on-board autonomy
  - Required Functions
    - » Action planning
    - » Execution Monitoring
    - » Fault diagnosis
- Model-based Reasoning via Model Checking
  - Environment and functions to carry out modeled as mathematical theories
  - Functions for autonomy (planning, monitoring and diagnosis) carried out as proof search (model checking)
  - Extensions of verification algorithms
- From off-line to on-line
  - can be useful for ground operation
  - carried out on board

Autonomous Systems

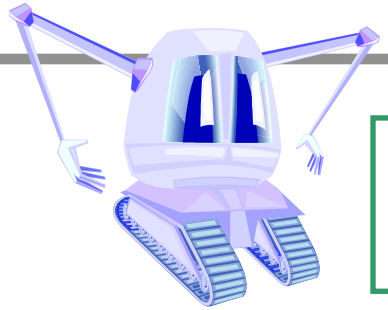
Model-Based Autonomy



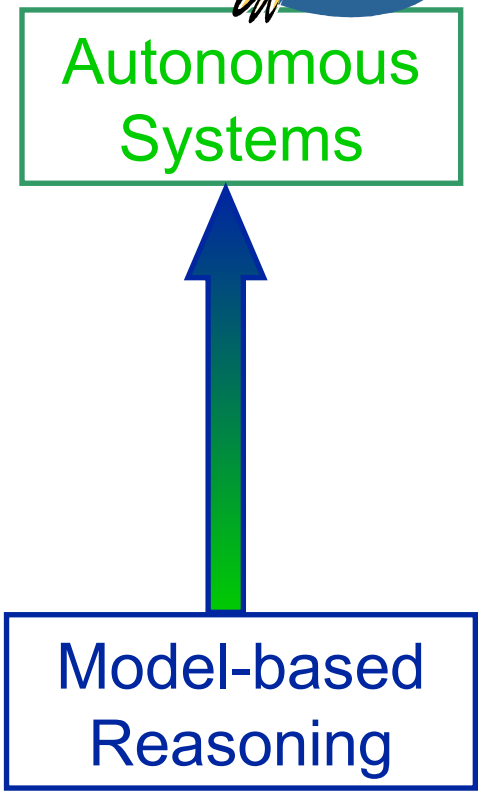
Autonomous Systems

- Autonomous systems
  - Example: remote space systems
- How can we program such systems?
  - Potential for on-board autonomy
  - Required Functions
    - » Action planning
    - » Execution Monitoring
    - » Fault diagnosis
- Model-based Reasoning via Model Checking
  - Environment and functions to carry out modeled as mathematical theories
  - Functions for autonomy (planning, monitoring and diagnosis) carried out as proof search (model checking)
  - Extensions of verification algorithms
- From off-line to on-line
  - can be useful for ground operation
  - carried out on board

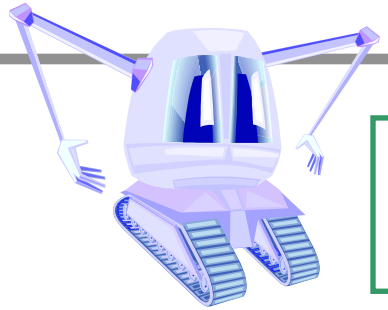
Model-Based Autonomy



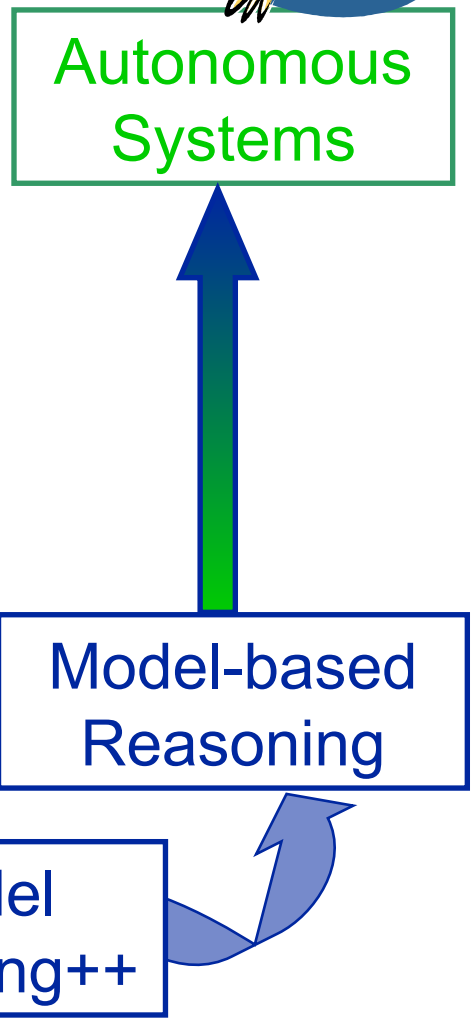
- Autonomous systems
  - Example: remote space systems
- How can we program such systems?
  - Potential for on-board autonomy
  - Required Functions
    - » Action planning
    - » Execution Monitoring
    - » Fault diagnosis
- Model-based Reasoning via Model Checking
  - Environment and functions to carry out modeled as mathematical theories
  - Functions for autonomy (planning, monitoring and diagnosis) carried out as proof search (model checking)
  - Extensions of verification algorithms
- From off-line to on-line
  - can be useful for ground operation
  - carried out on board



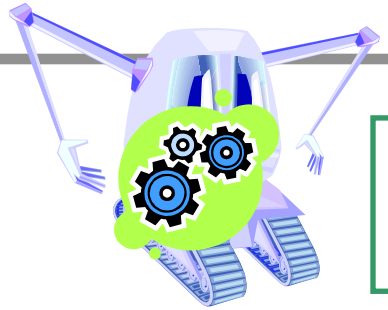
Model-Based Autonomy



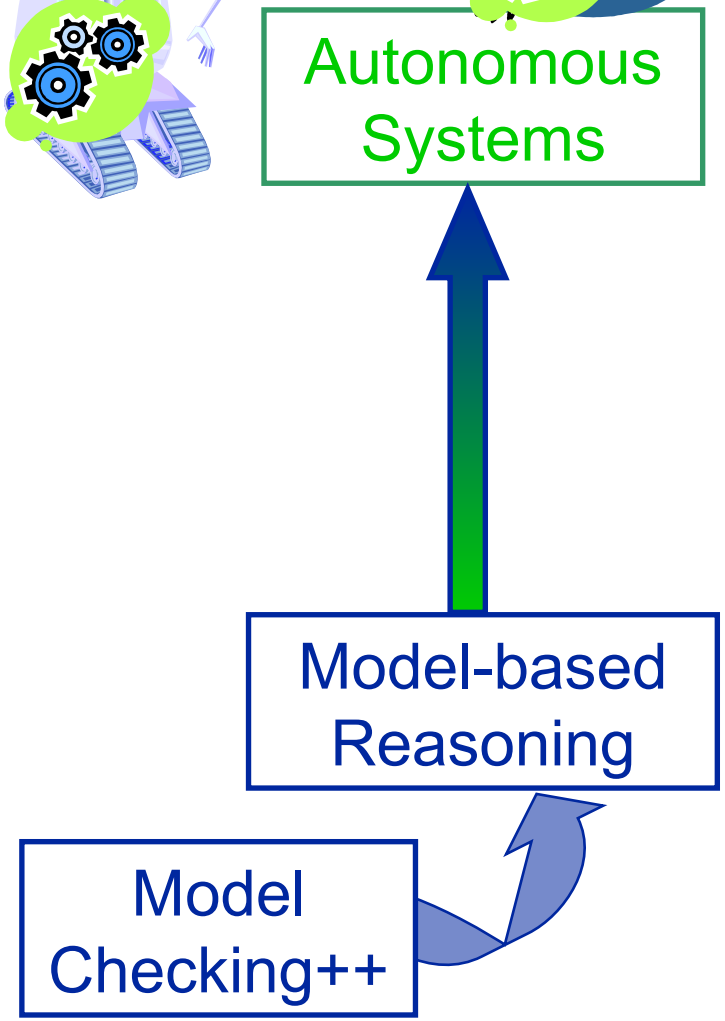
- Autonomous systems
  - Example: remote space systems
- How can we program such systems?
  - Potential for on-board autonomy
  - Required Functions
    - » Action planning
    - » Execution Monitoring
    - » Fault diagnosis
- Model-based Reasoning via Model Checking
  - Environment and functions to carry out modeled as mathematical theories
  - Functions for autonomy (planning, monitoring and diagnosis) carried out as proof search (model checking)
  - Extensions of verification algorithms
- From off-line to on-line
  - can be useful for ground operation
  - carried out on board



Model-Based Autonomy



- Autonomous systems
  - Example: remote space systems
- How can we program such systems?
  - Potential for on-board autonomy
  - Required Functions
    - » Action planning
    - » Execution Monitoring
    - » Fault diagnosis
- Model-based Reasoning via Model Checking
  - Environment and functions to carry out modeled as mathematical theories
  - Functions for autonomy (planning, monitoring and diagnosis) carried out as proof search (model checking)
  - Extensions of verification algorithms
- From off-line to on-line
  - can be useful for ground operation
  - carried out on board



Results in Model-based Reasoning

- The research line in Planning as Model Checking
  - Several top publications:
    - » 3 Journal of Artificial Intelligence
    - » 4 IJCAI
  - Citation in MIT book of abstracts
  - Collaboration with NASA Ames Research Center
  - Recent award ICAPS'08 most influential paper in the last decade
- The MBP planner
  - applied also in software and hardware synthesis
- The "On-board Model Checking" project
  - Invitation to Tender by European Space Agency
  - FBK-irst in team with Thales-Alenia Space
  - Theme: model-based reasoning for autonomous systems
    - » Model based planning + execution + monitoring
    - » Embedded platforms

Overview of the ARE Architecture

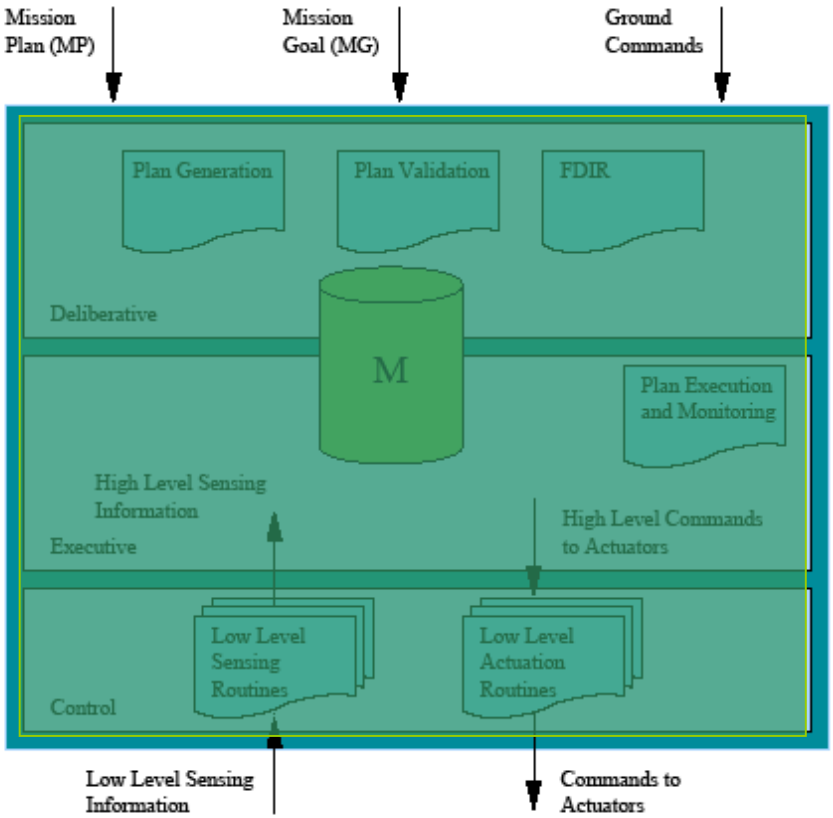
Marco Roveri

The proposed solution

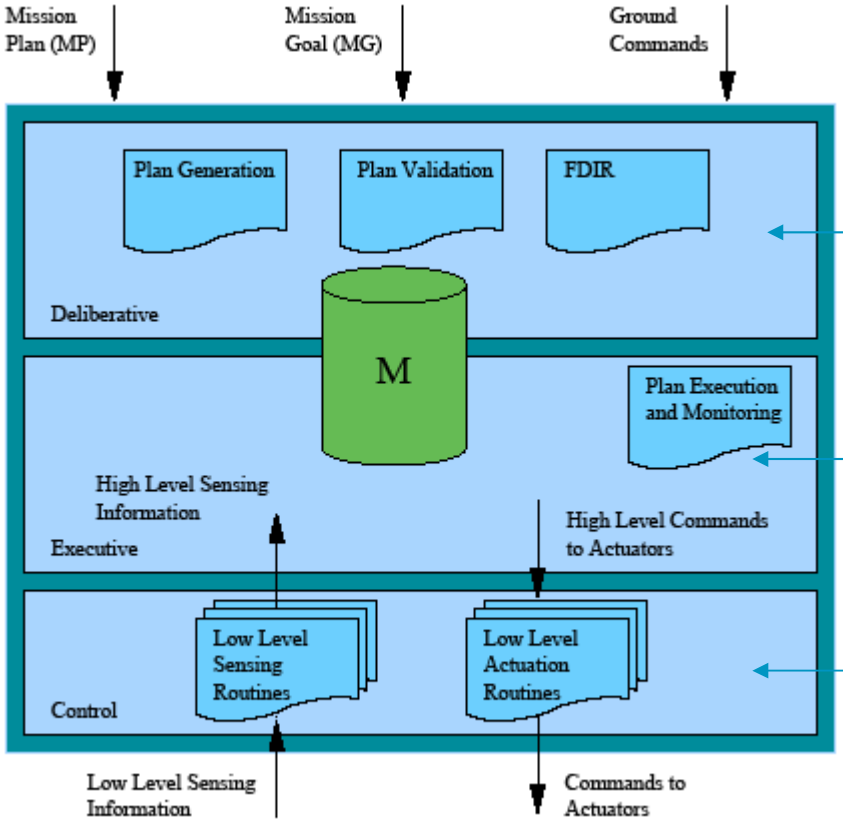
- Autonomy as a mix of cooperating functions
  - Plan generation, plan validation
  - Execution and Monitoring
  - FDIR
- A layered approach
  - Deliberative layer
  - Executive layer
  - Control layer
- A generic reasoner working on a model
  - Same model shared between the cooperating functions

Autonomy Architecture: BB view

- High-level Inputs
  - mission plans
  - mission goals
  - ground commands
- High-level Outputs
  - logs, inspected status
- Low level Inputs
  - signals from sensors
- Low level Outputs
  - commands to actuators



ARE high level architecture

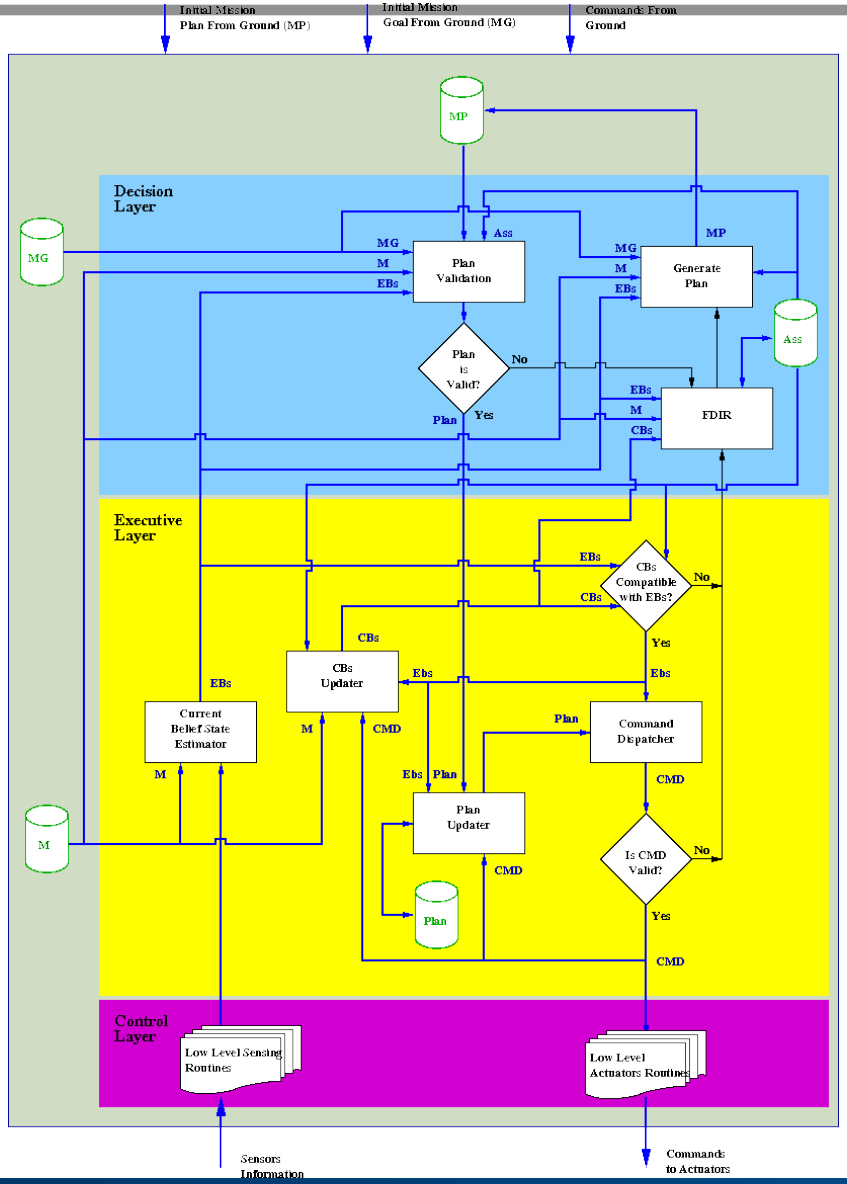


DELIBERATIVE LAYER: provides facilities for goal-driven planning, plan validation, and system-level FDIR.

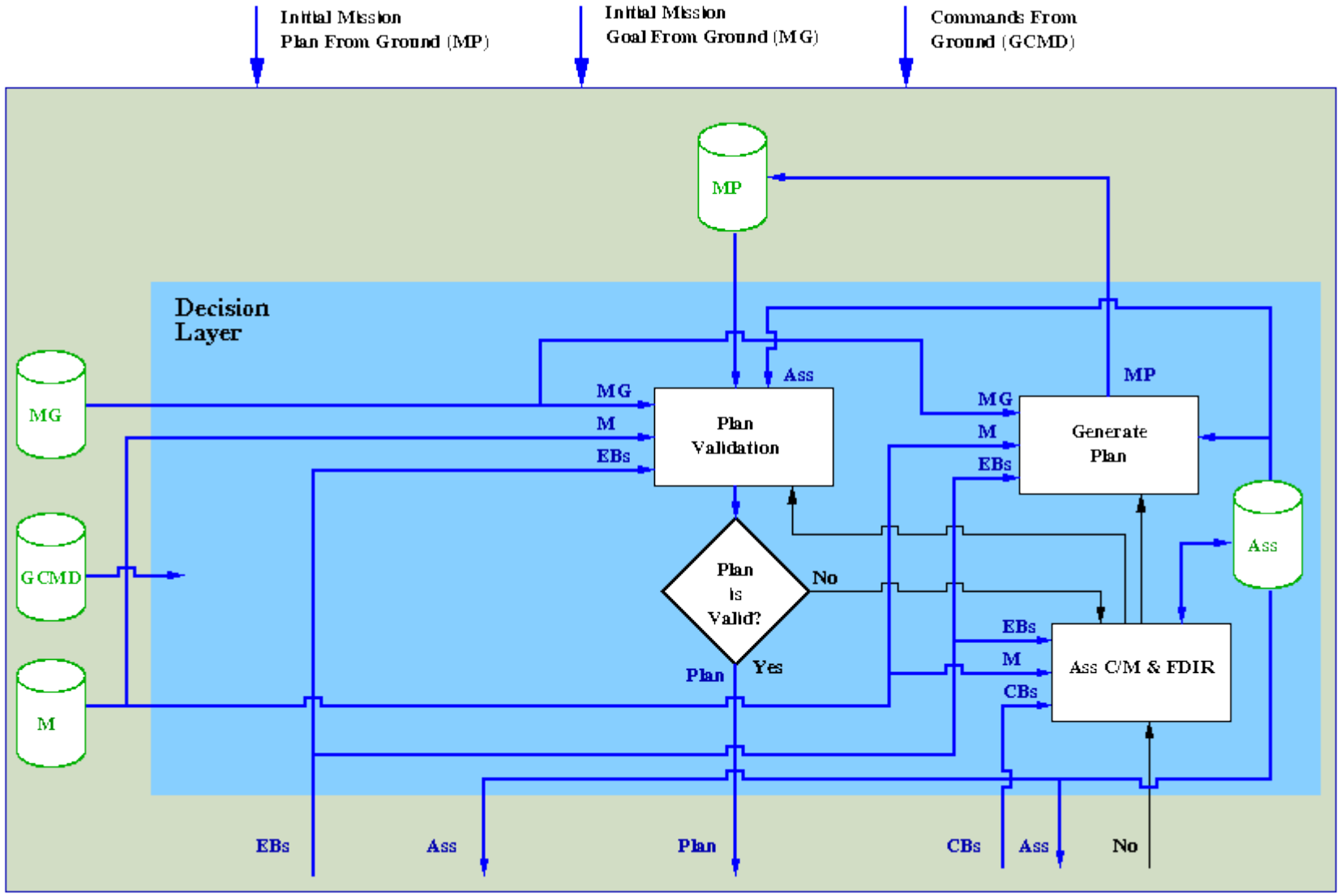
EXECUTIVE LAYER: provides facilities to execute and monitor the correct execution of the current mission plan.

CONTROL LAYER: provides low level interactions with the controlled system (sensor acquisition and commands to actuators sending).

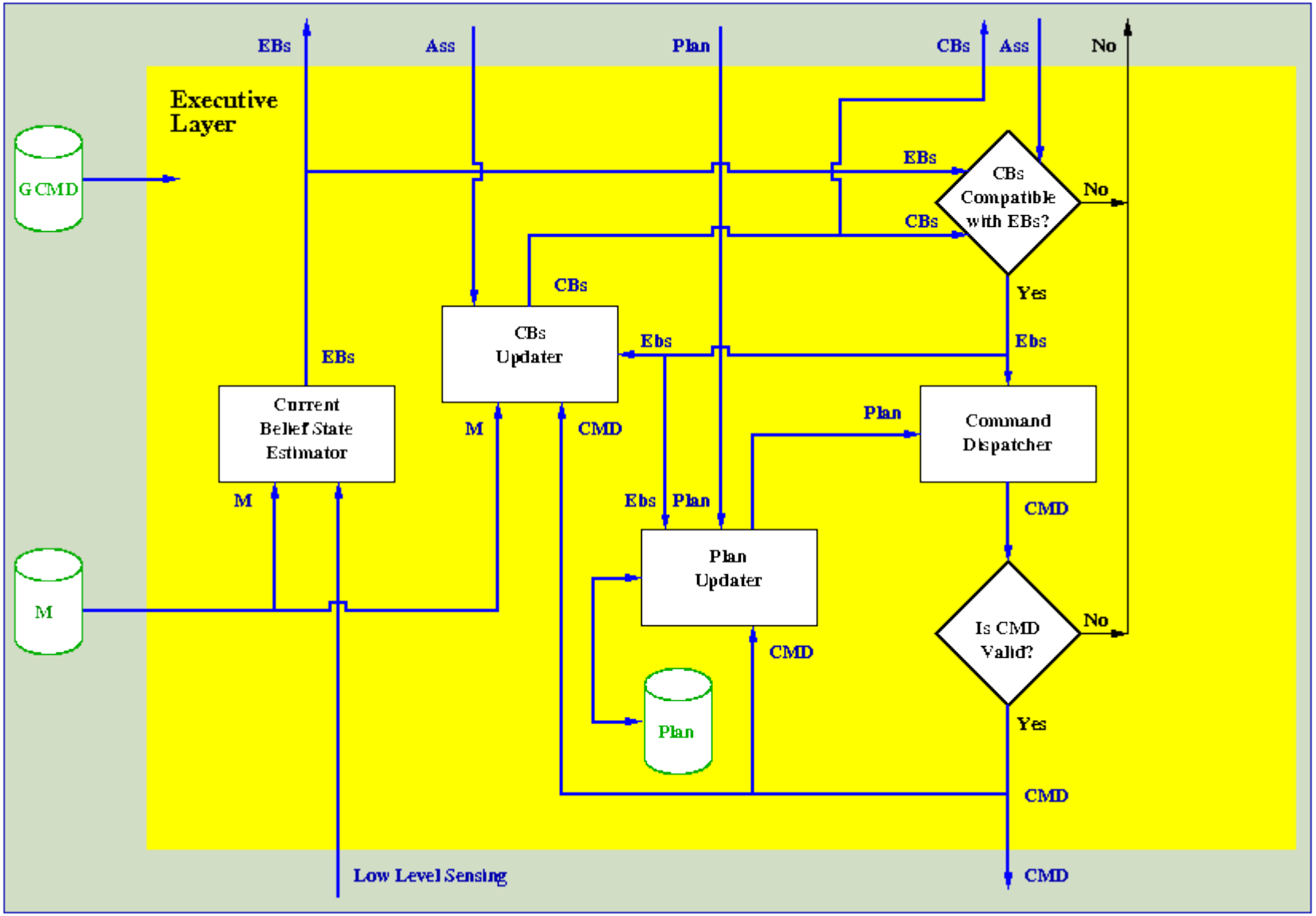
ARE architecture overview



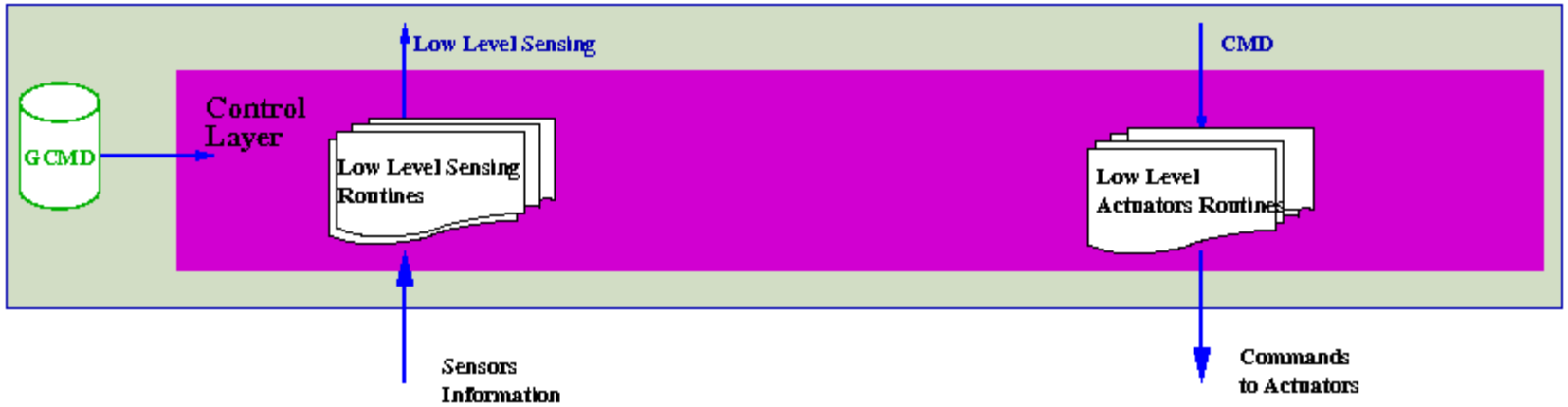
ARE Decision Layer



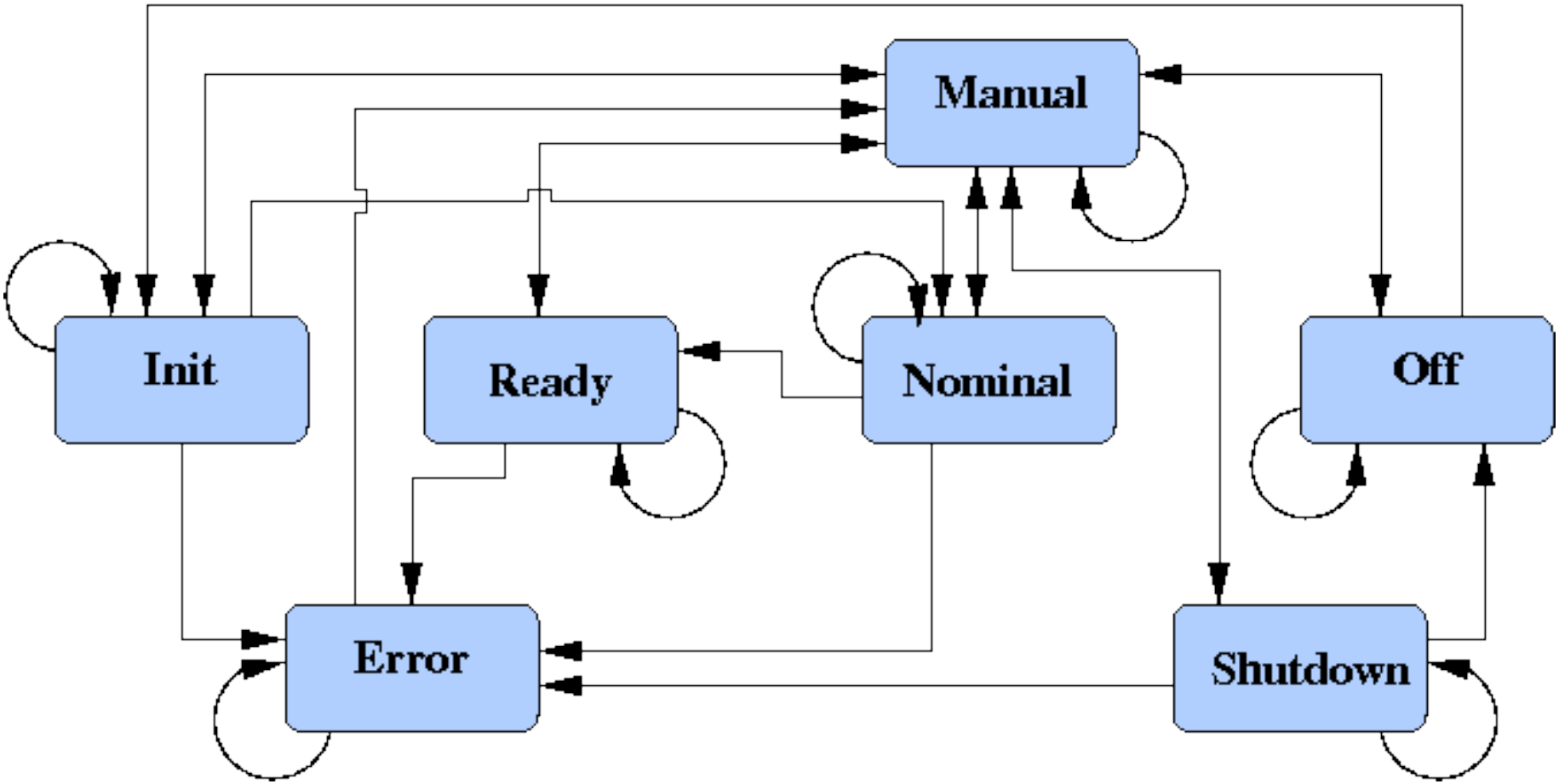
ARE Executive Layer



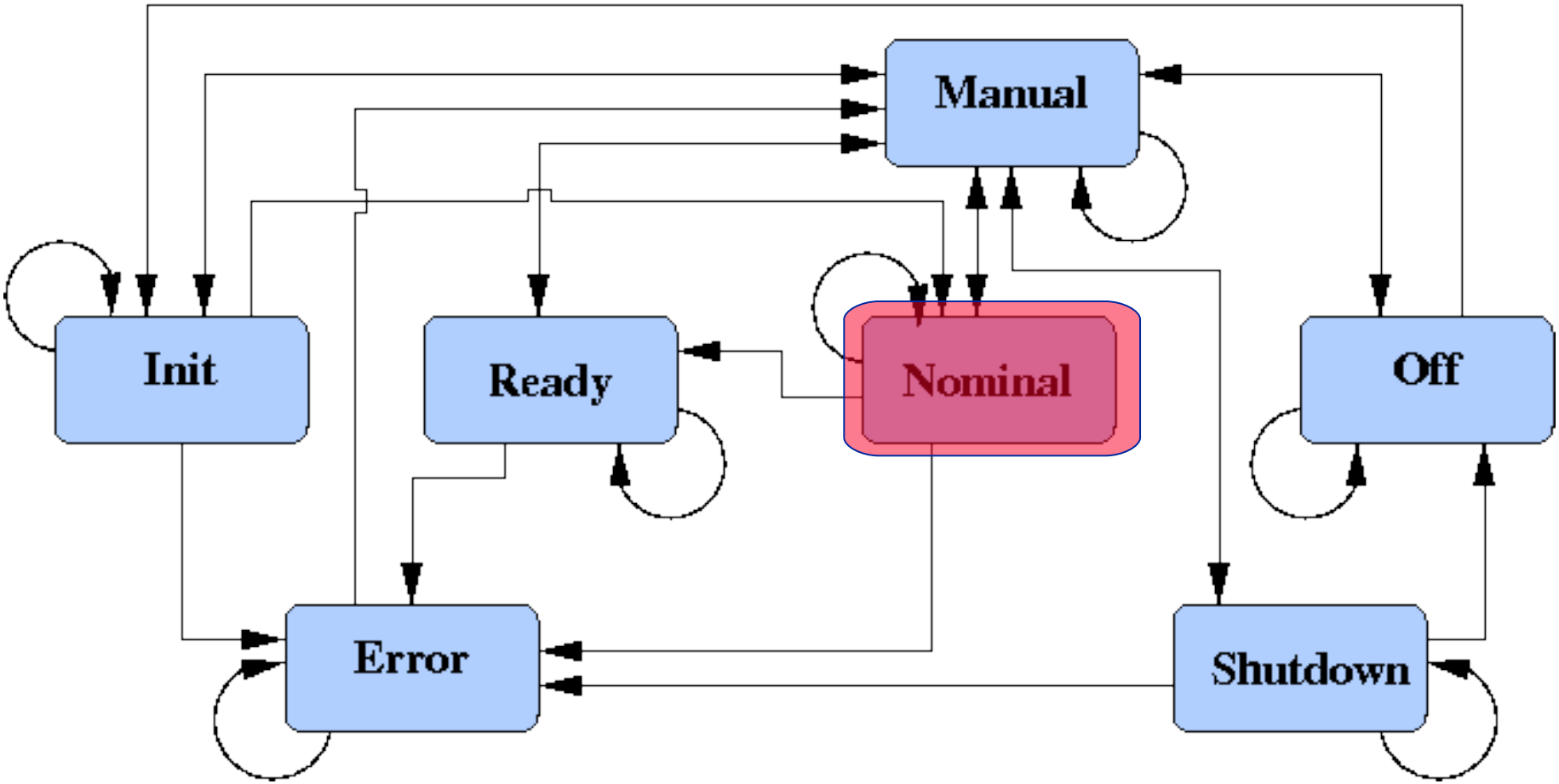
ARE Control Layer



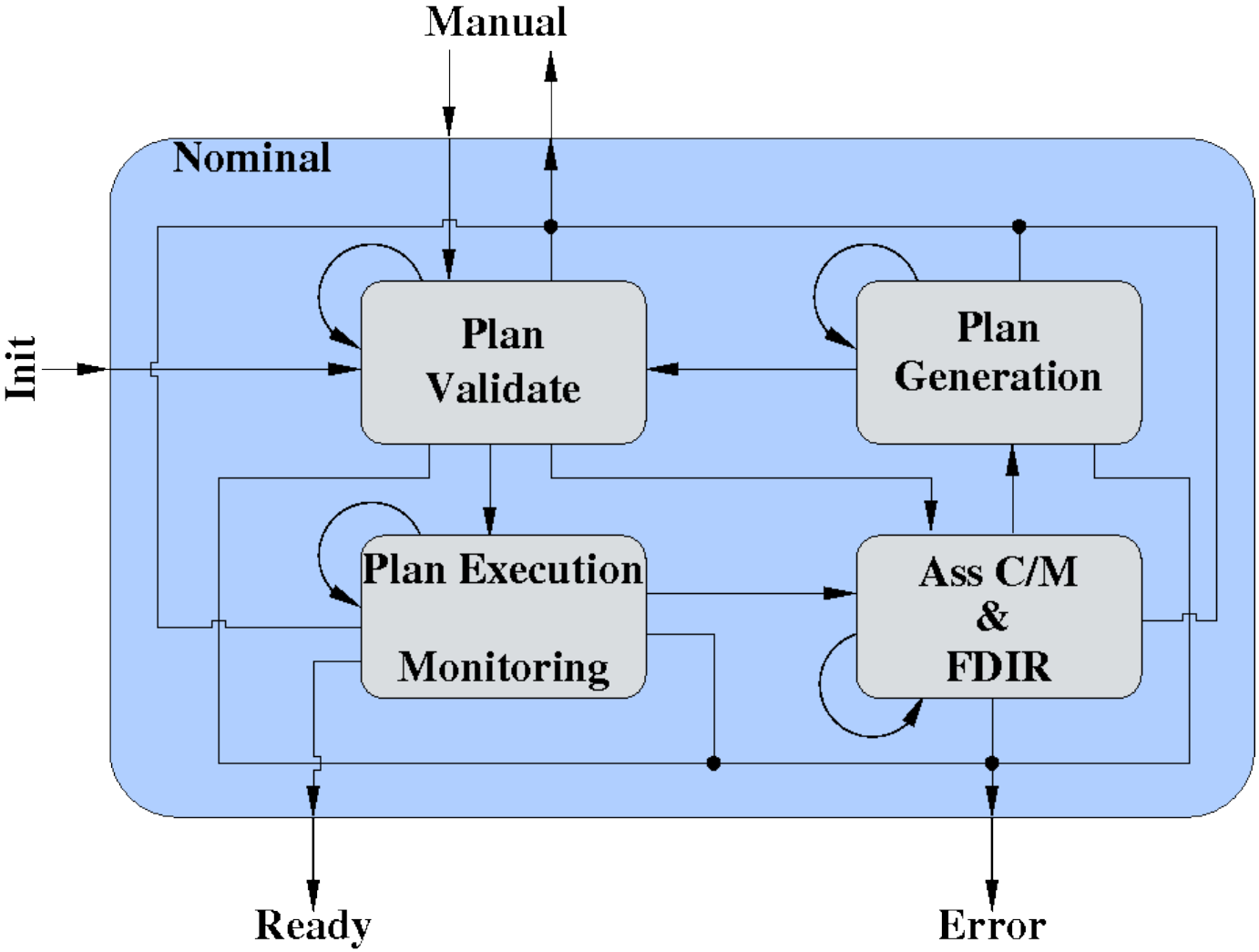
ARE Top Level FSM



ARE Top Level FSM



ARE FSM: the Nominal state



FONDAZIONE
BRUNO KESSLER

ThalesAlenia
Space

The ARE Formal Framework

Alessandro Cimatti

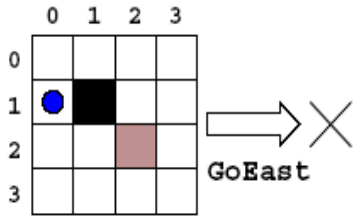
ARE Formal Framework

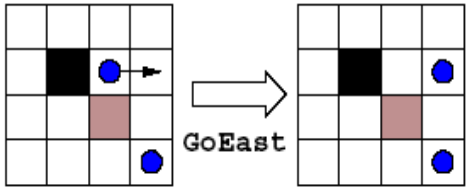
- The Planning as Model Checking framework
  - Possible non-determinism in action effects
  - Limited sensing: partially observable domain
  - Modeling run-time uncertainty on plant status
  - Conditional plans
- Extension 1: *Assumptions* under which the controlled plant is operated
- Extension 2: Model of *Resources* for plan validation and run-time monitoring
- Result:
  - generation of conditional strong plans
  - annotated for run-time monitoring of assumptions …
  - … and of resource consumption

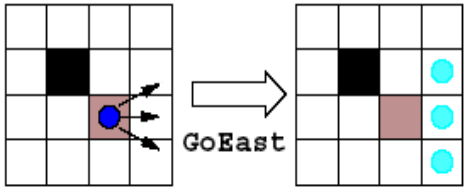
The Formal Model

- Q is a set of propositions
- R is a set of resources
- S is finite set of states
- A is a finite set of actions
- $O \subseteq Q$ is a set of observations
- $F \subseteq Q$ is a set of fault bits
- $T \subseteq S \times A \times S$
  - The transition *relation*
  - Applicability
  - Several possible outcomes
- $X \subseteq O \times S$
  - The observation relation, associating to a state the observation compatible with that state.
- $Lab : S \rightarrow 2Q$
  - Labeling function
- $Res : S \rightarrow (R \rightarrow \Re 2)$
  - Resource association function
  - for each resource in a state a Low value ($Res(s)(r)^{Low}$) and an High value ($Res(s)(r)^{High}$)
- $Ass \subseteq S$
  - Assumptions under which the plant is operated

An Example







- Simple robot navigation domain
- states: 00, .., 03, 10, 12, …
- actions: GoNorth, GoSouth, GoEast, GoWest
- applicability: R(01, GoEast) = { }
- determinism: R(33, GoEast) = { 33 }
- nondeterminism: R(22, GoEast) = { 31, 32, 33 }

Reasoning in Belief Space

- Dealing with uncertainty
  - several different states may be compatible with currently available information
  - collect indistinguishable states into a set called Belief State
- Belief state $\emptyset \neq BS \subseteq S$
  - Emptiness denotes inconsistency in representation
- Actions in belief space
  - applicability conditions must hold in all states
  - result belief state is set of all possible successors
- Observations “split” belief states
  - limit BS to the states compatible with observation

Dealing with Resources

- For run-time plan monitoring and for plan validation we use two additional functions:
  - A resource estimation function (REF) that associates to a belief state BS $\subseteq$ S and to a resource r$\in$R a pair <m,M>$\in \Re^2$

    $$REF : 2^S \rightarrow (R \rightarrow \Re^2)$$

  - A resource consumption estimation function (RCEF) that associates to a BS $\subseteq$ S, to an action $\alpha \in$ A, and to a resource r$\in$R a pair <m,M>$\in \Re^2$ corresponding to the min and max resource consumed by executing action α in any state s$\in$BS

    $$RCEF : 2^S \times A \rightarrow (R \rightarrow \Re)$$

Resources in Belief Space

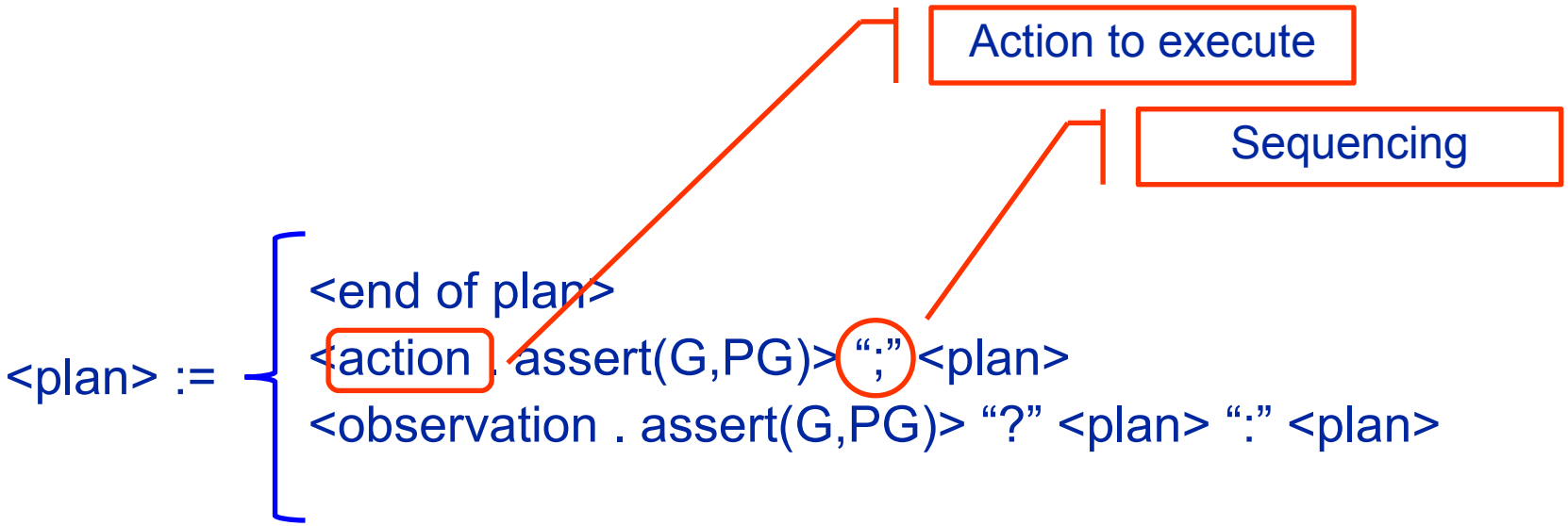
- REF(BS)(r) := <m,M>
  - $m := \min\{Res(s)(r)^{Low} \mid s \in BS\}$
  - $M := \max\{Res(s)(r)^{High} \mid s \in BS\}$

- RCEF(BS, α)(r) := <m,M>
  - $m := \min\{Res(s)(r)^{Low} - Res(s')(r)^{High} \mid s \in BS, (s,\alpha,s') \in T\}$
  - $M := \max\{Res(s)(r)^{High} - Res(s')(r)^{Low} \mid s \in BS, (s,\alpha,s') \in T\}$

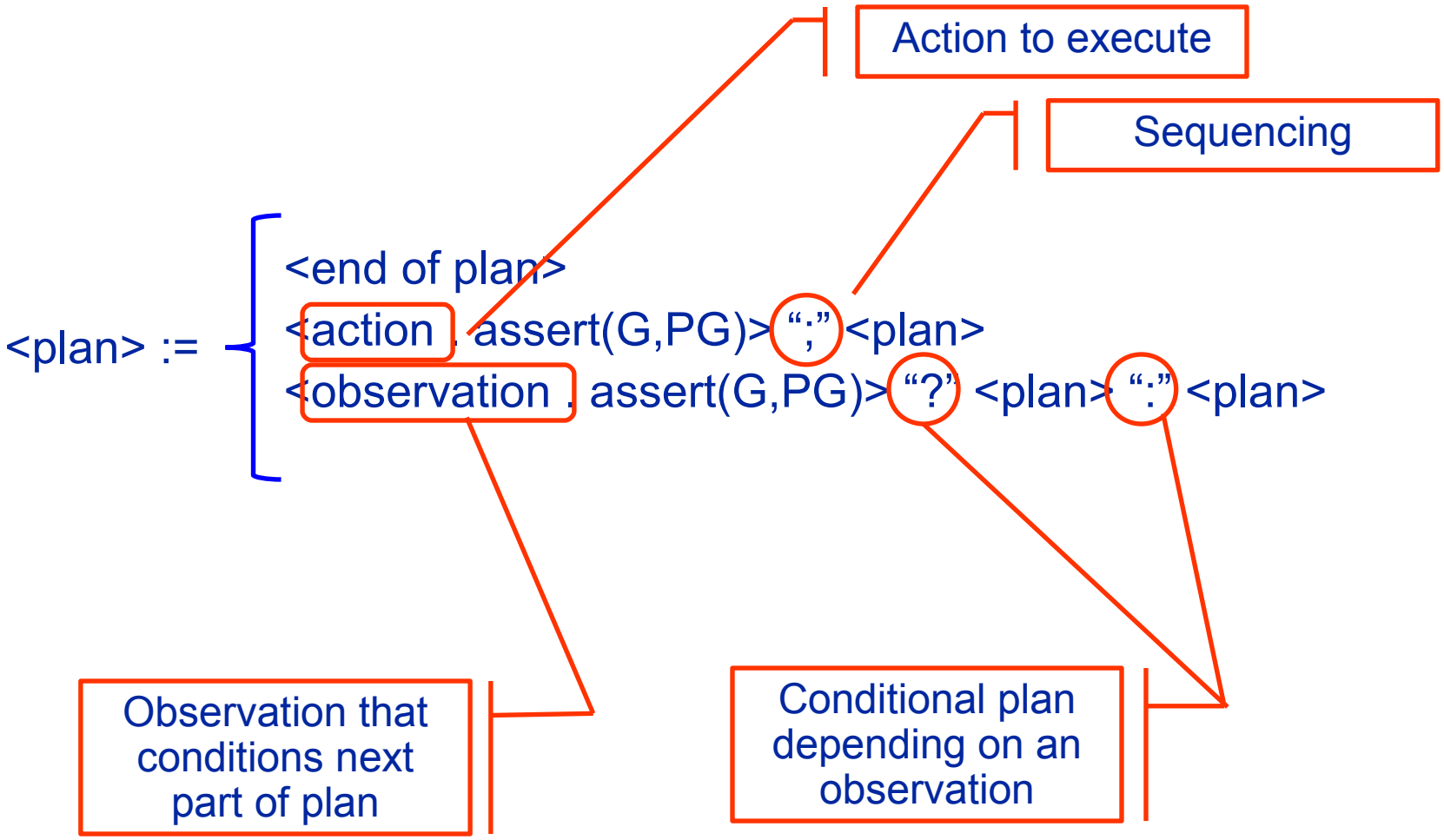
Mission Plans

<plan> :=
- <end of plan>
- <action . assert(G,PG)> “;” <plan>
- <observation . assert(G,PG)> “?” <plan> “:” <plan>

Mission Plans



Mission Plans

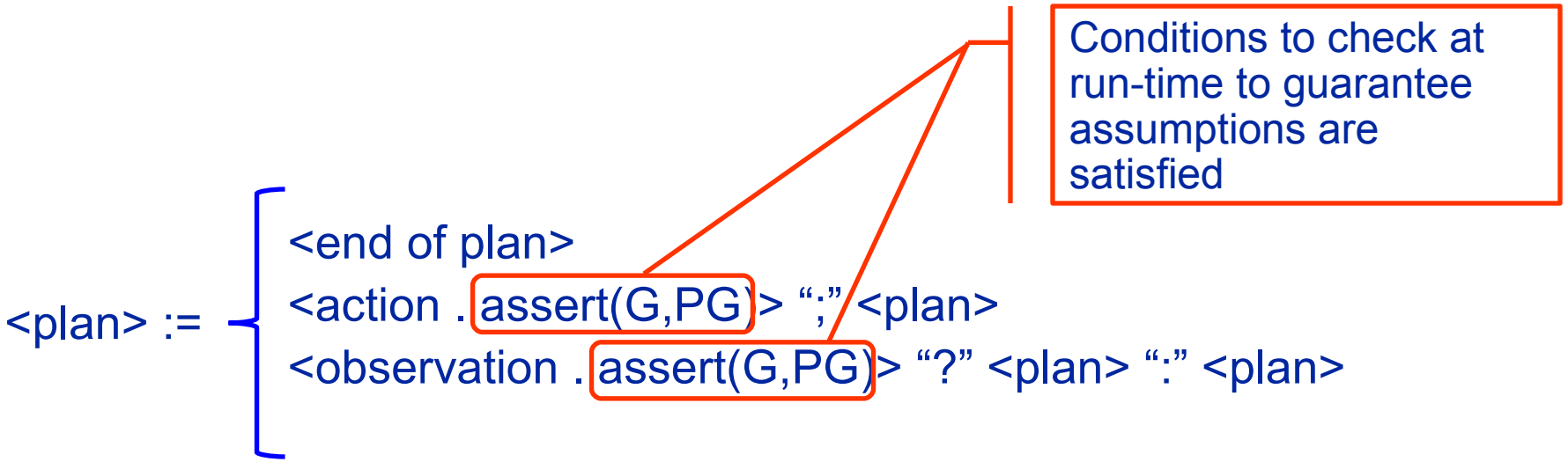


Mission Plans (II)

<plan> :=
- <end of plan>
- <action . assert(G,PG)> “;” <plan>
- <observation . assert(G,PG)> “?” <plan> “:” <plan>

- G $\subseteq$ S is a set of states in which the assumption Ass holds
- G $\subseteq$ PG $\subseteq$ S is a set of states in which the assumption Ass may be violated, but that are indistinguishable from states in G.

Mission Plans (II)



- $G \subseteq S$ is a set of states in which the assumption Ass holds
- $G \subseteq PG \subseteq S$ is a set of states in which the assumption Ass may be violated, but that are indistinguishable from states in G.

Plan Execution: Exec[P](S)

- Recursively defined on the plan structure
  - Exec[<end of plan>](BS) = BS
  - Exec[< α >;P](BS) = $\emptyset$ if α is not applicable in BS
  - Exec[< α >;P](BS) = BS' if α is applicable in BS
    - » BS' = {s'$\in$ S| s$\in$ BS and (s,α,s')$\in$ T}
  - Exec[o ? P_T : P_F](BS) = $\emptyset$ if
    - » Exec[P_T](BS[o,T]) = $\emptyset$ or Exec[P_F](BS[o,F]) = $\emptyset$
  - Exec[o ? P_T : P_F](BS) = BS'
    - » BS' = Exec[P_T](BS[o,T]) $\cup$ Exec[P_F](BS[o,F]) if Exec[P_T](BS[o,T]) $\neq \emptyset$ and Exec[P_F](BS[o,F]) $\neq \emptyset$

Plan Execution with Assumptions

- $Exec[P](\langle S_g, S_{pg}\rangle) = \langle F_g, F_{pg}\rangle$
- S_{pg} is the set of states indistinguishable from S_g
  - $S_{pg} = IND(S_g)$
- F_g, F_{pg} obtained by progressing the S_g, S_{pg} following the plan similarly to the case without assumption
  - Action applicability checked w.r.t S_{pg}
  - Observations split both S_g, S_{pg}

Plan Execution with Resources

- Resources associated to S_g and to S_{pg}
- Resources progressed from initial resources according to
  - plan structure,
  - resource cost estimation function
- If resources of current S_{pg} less than limit R_{min} then return $\langle \emptyset, \emptyset \rangle$

Plan Validity

- $\emptyset \neq Init \subseteq S$
  - Set of all possible initial states (Initial belief state)
- $\emptyset \neq Goal \subseteq S$
  - Set of all goal states
- Plan P is valid (is a strong solution) iff
  - $Exec[P](<Init,IND(Init)>) \neq <\emptyset, \emptyset>$
  - $Exec[P](<Init,IND(Init)>) = <F_g,F_{pg}>$ and $F_{pg} \subseteq Goal$

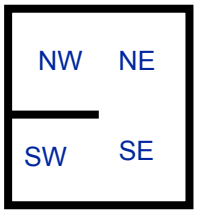
Plan Generation Algorithm

- Forward And-Or search in belief space
- Node Expansion
  - OR branching
    - » simulate effect of action execution
  - AND branching
    - » simulate effect of observation
- Nodes tagged as
  - success if contained in goal, or if descendent success
  - failure if no action possible or all descendents are failure due to loopbacks

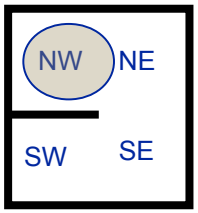
Plan Generation Algorithm (II)

- Beliefs pruned according to assumptions
- In parallel with the current belief, we also progress two *"monitor-beliefs"*, S_g and S_{pg} that represent uncertainty w.r.t. assumption status.
- We prune monitor-beliefs using sensing, until no more uncertainty.
- General, domain-independent heuristic guidance used
- Resource consumption currently disregarded during planning
  - could be used to prune resource-inconsistent branches

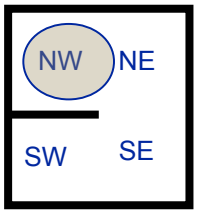
An example

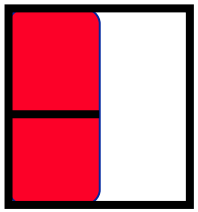


An example

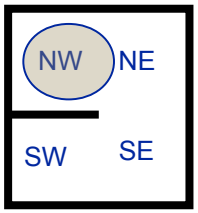


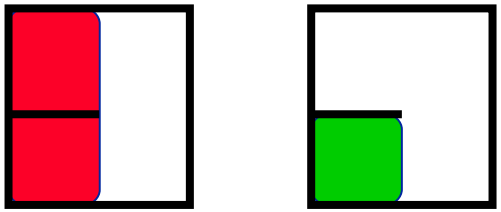
An example



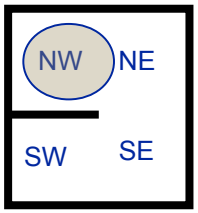


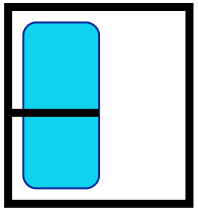
An example

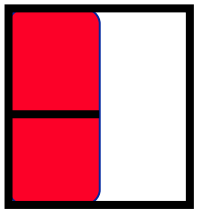


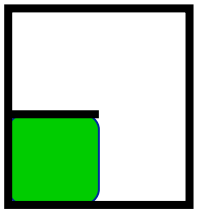


An example

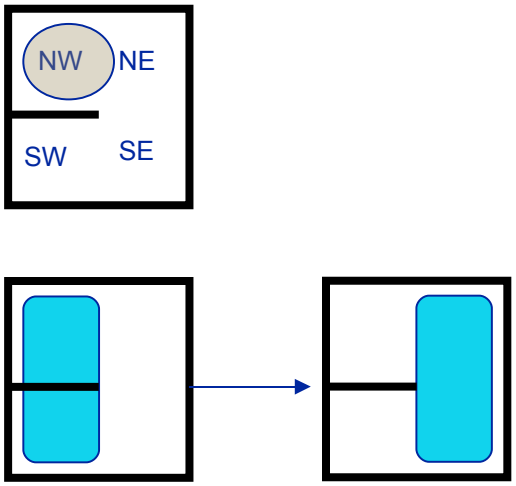


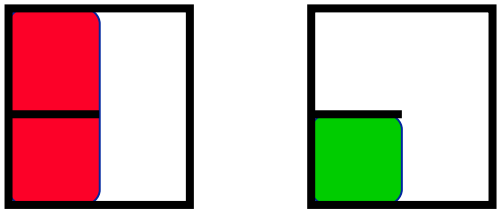




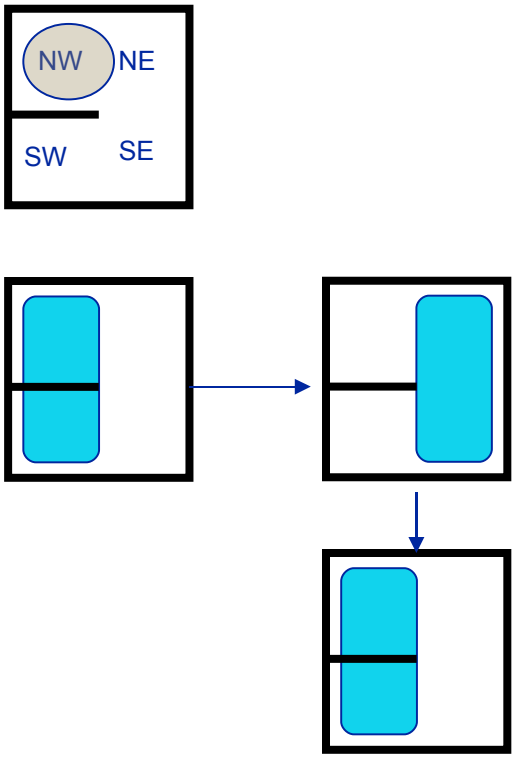


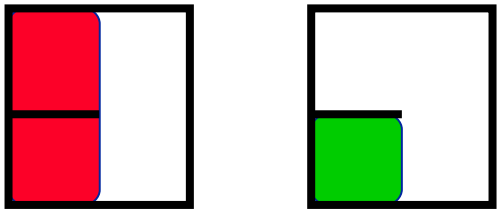
An example



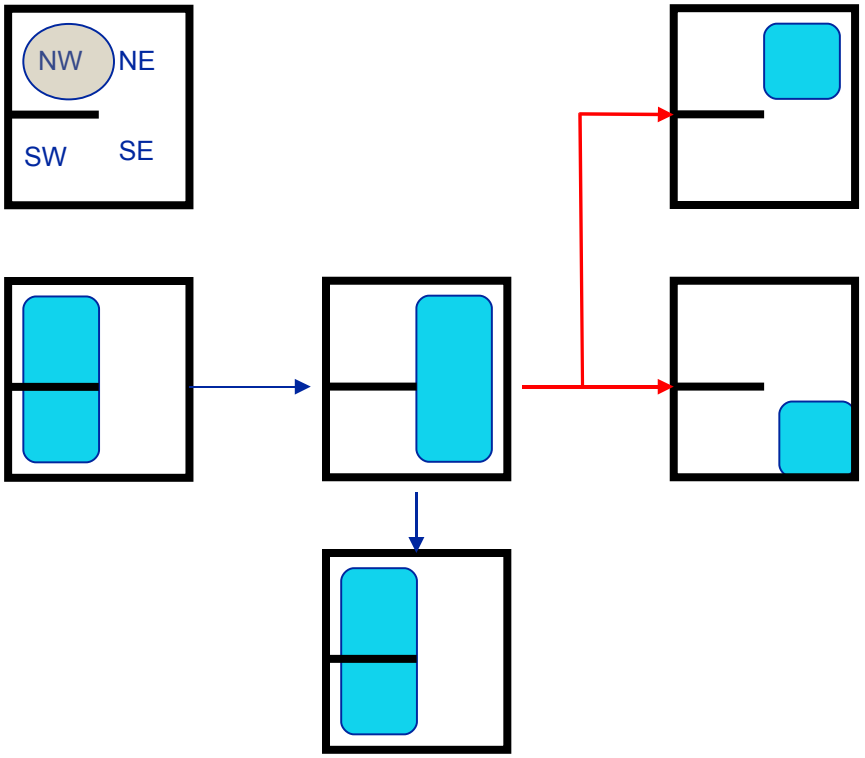


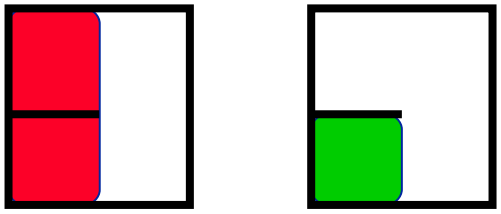
An example



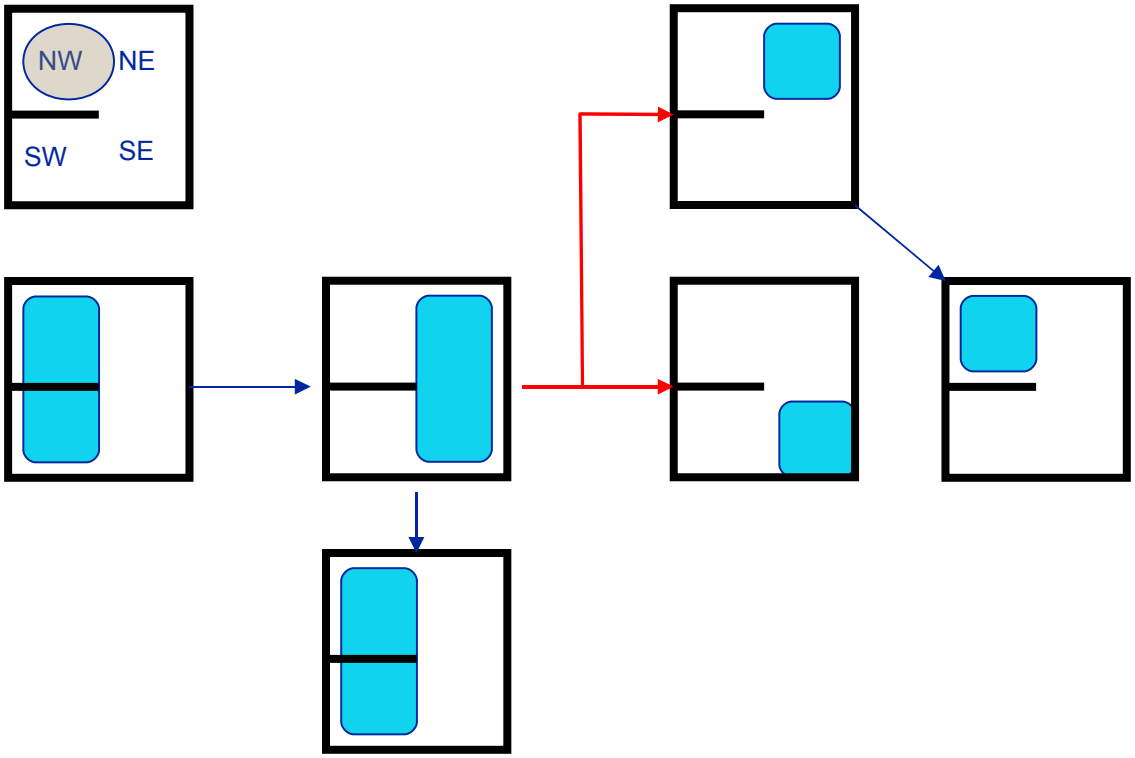


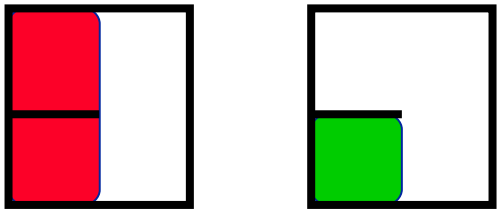
An example



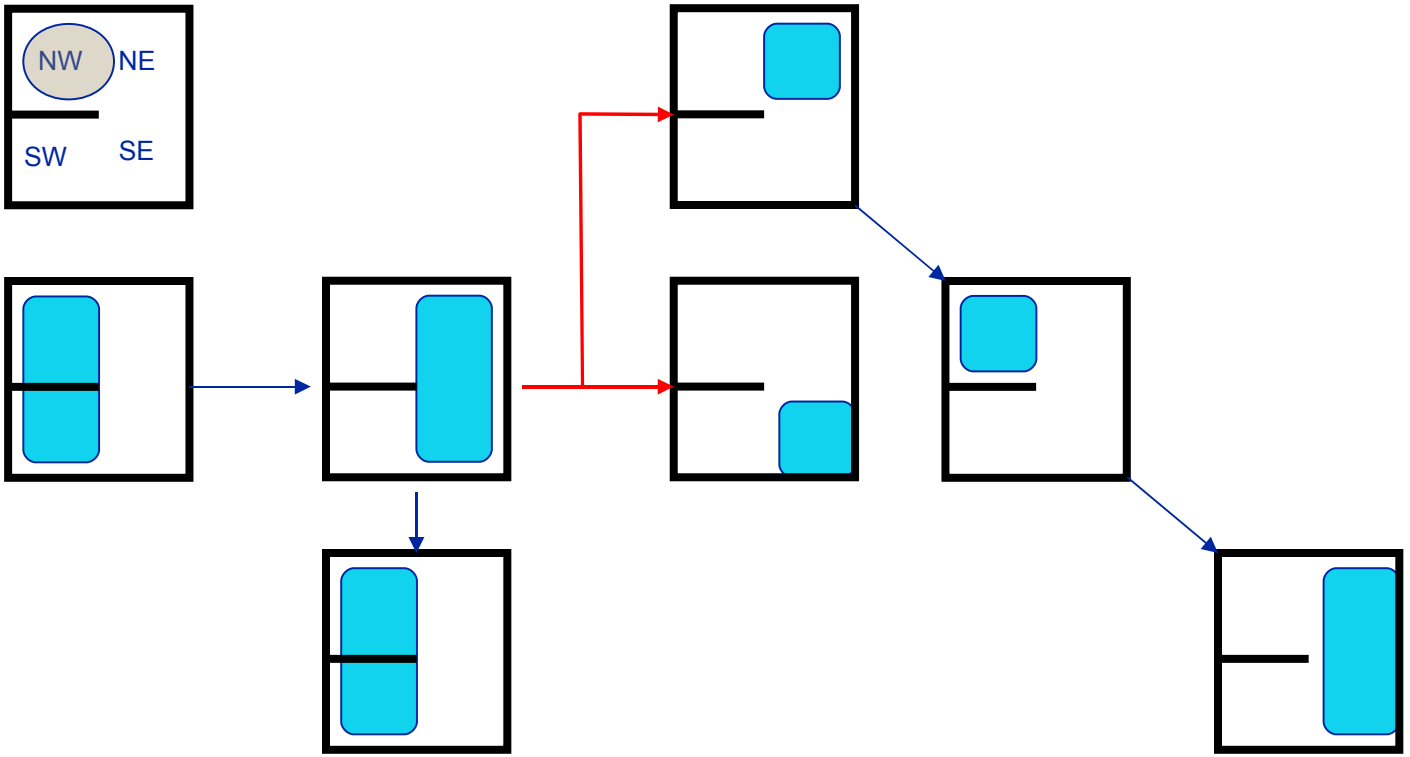


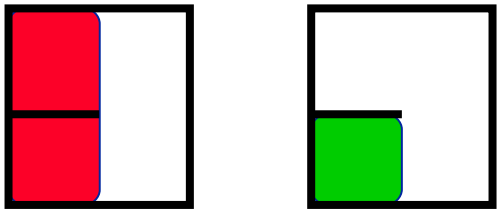
An example



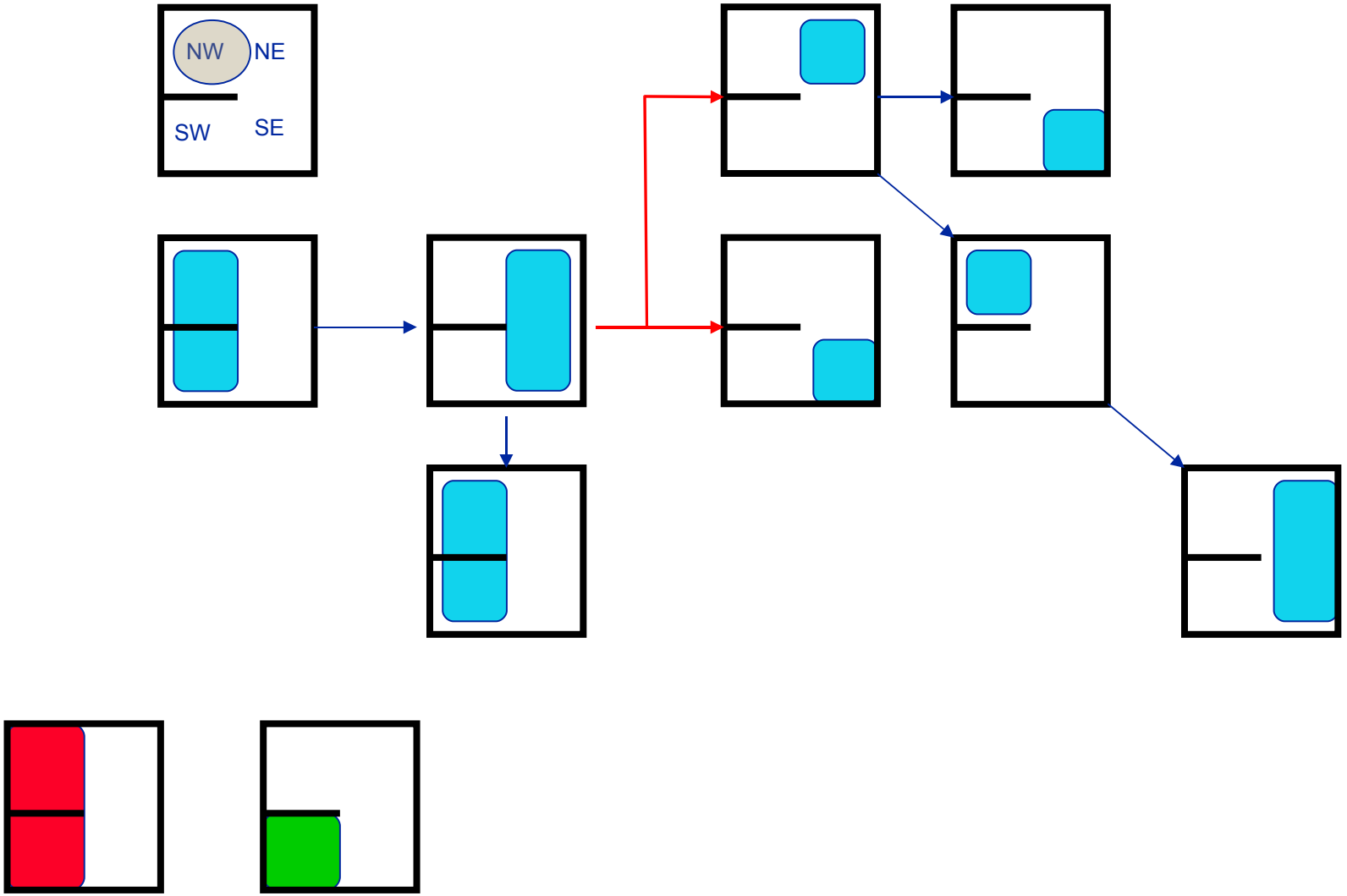


An example

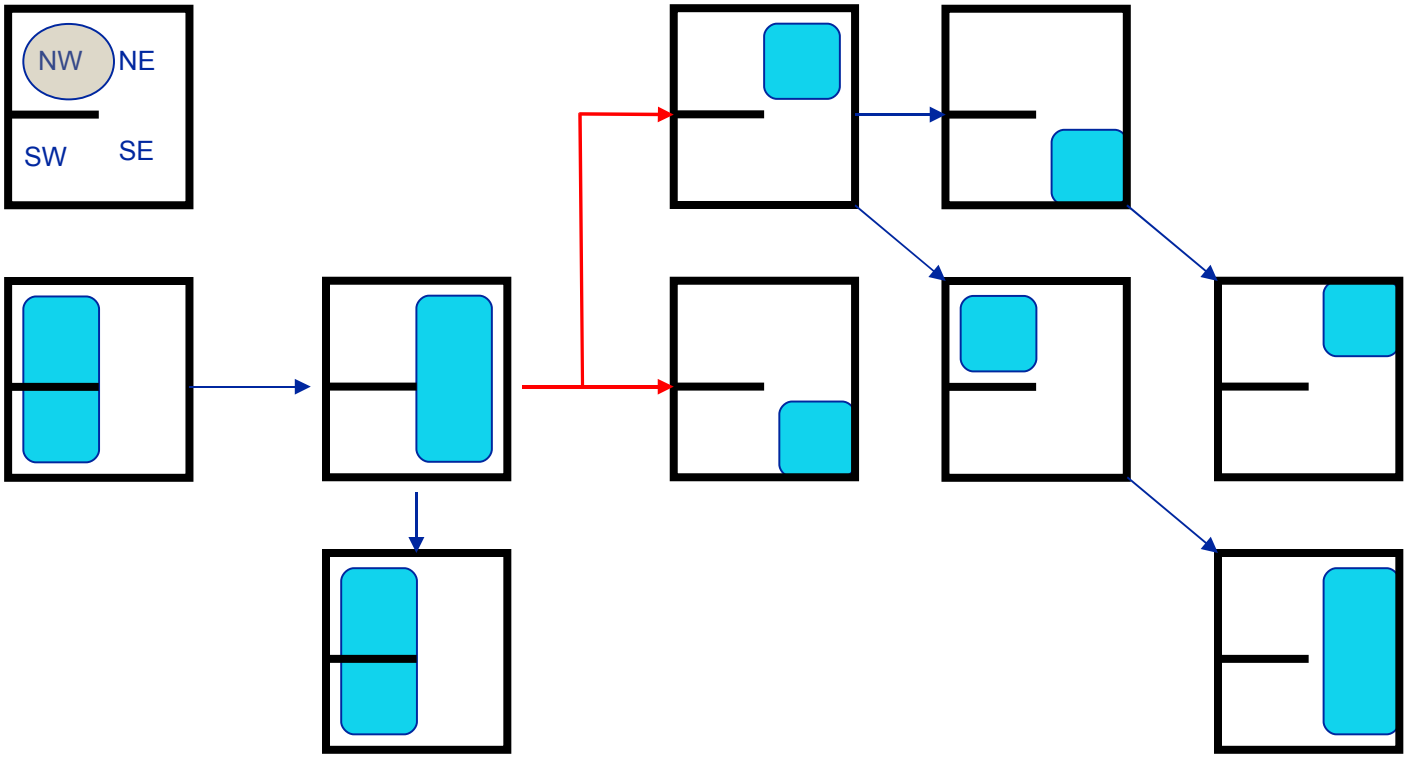


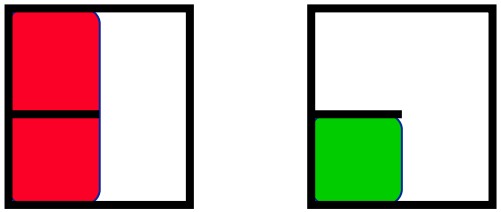


An example

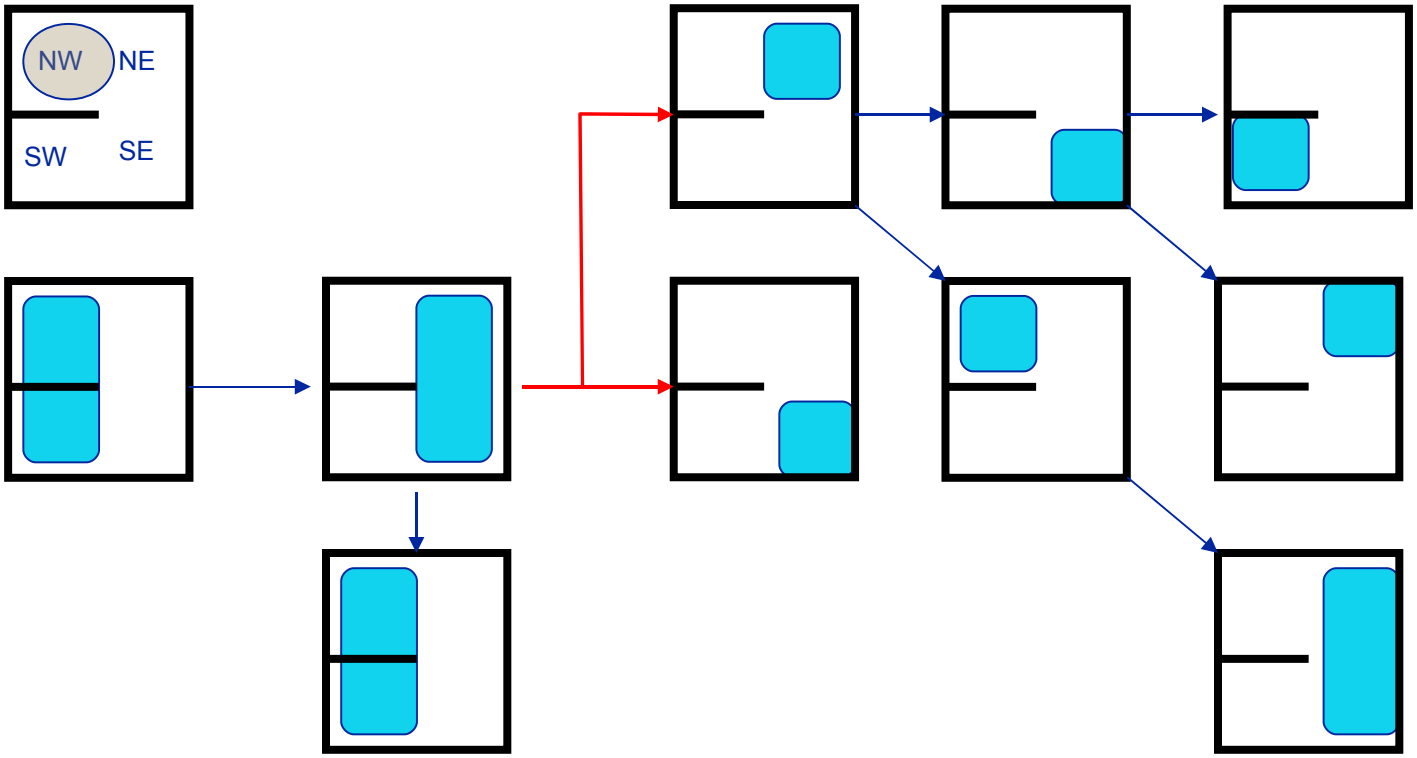


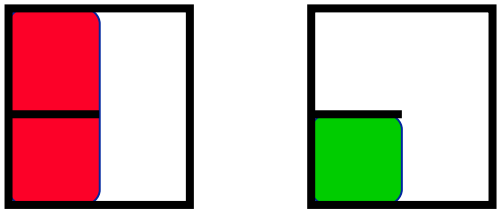
An example



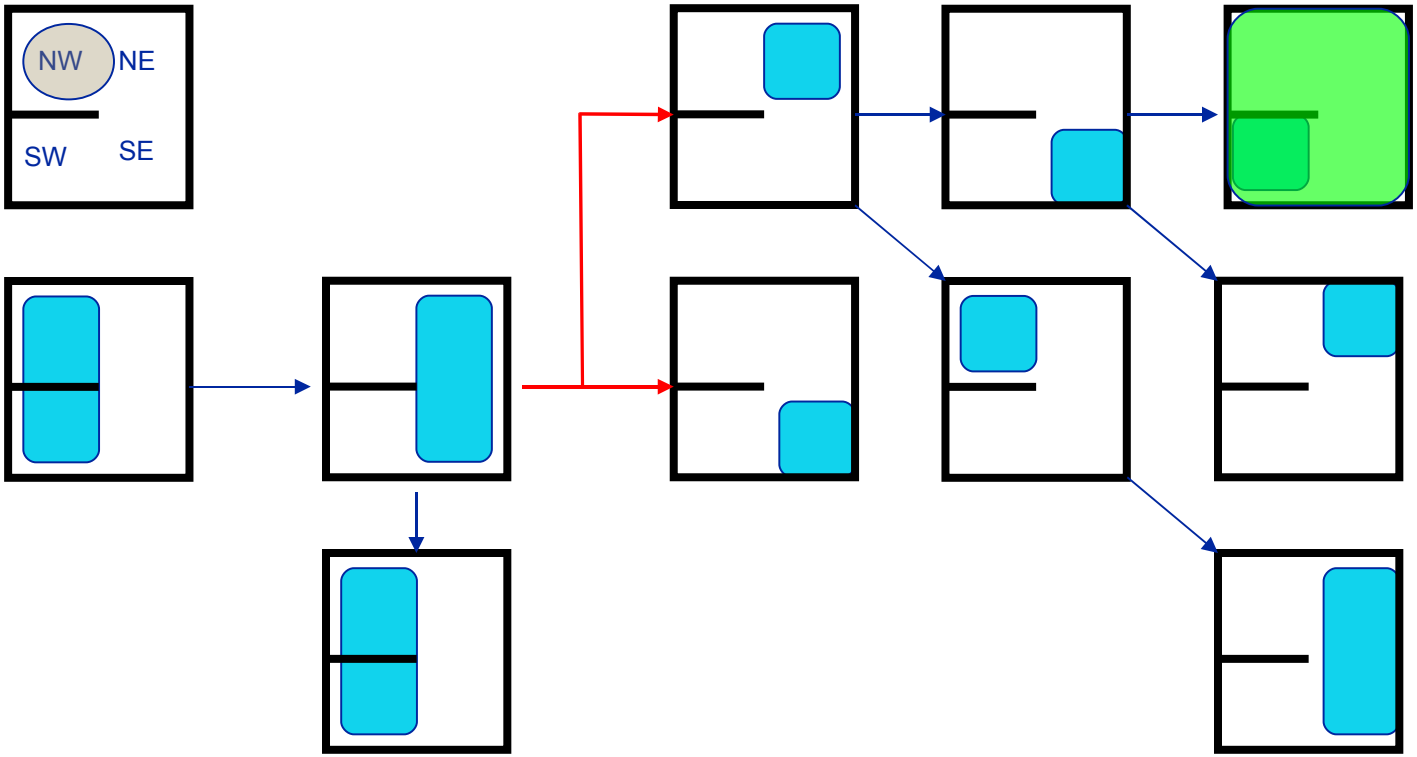


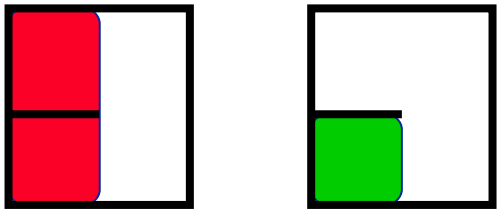
An example



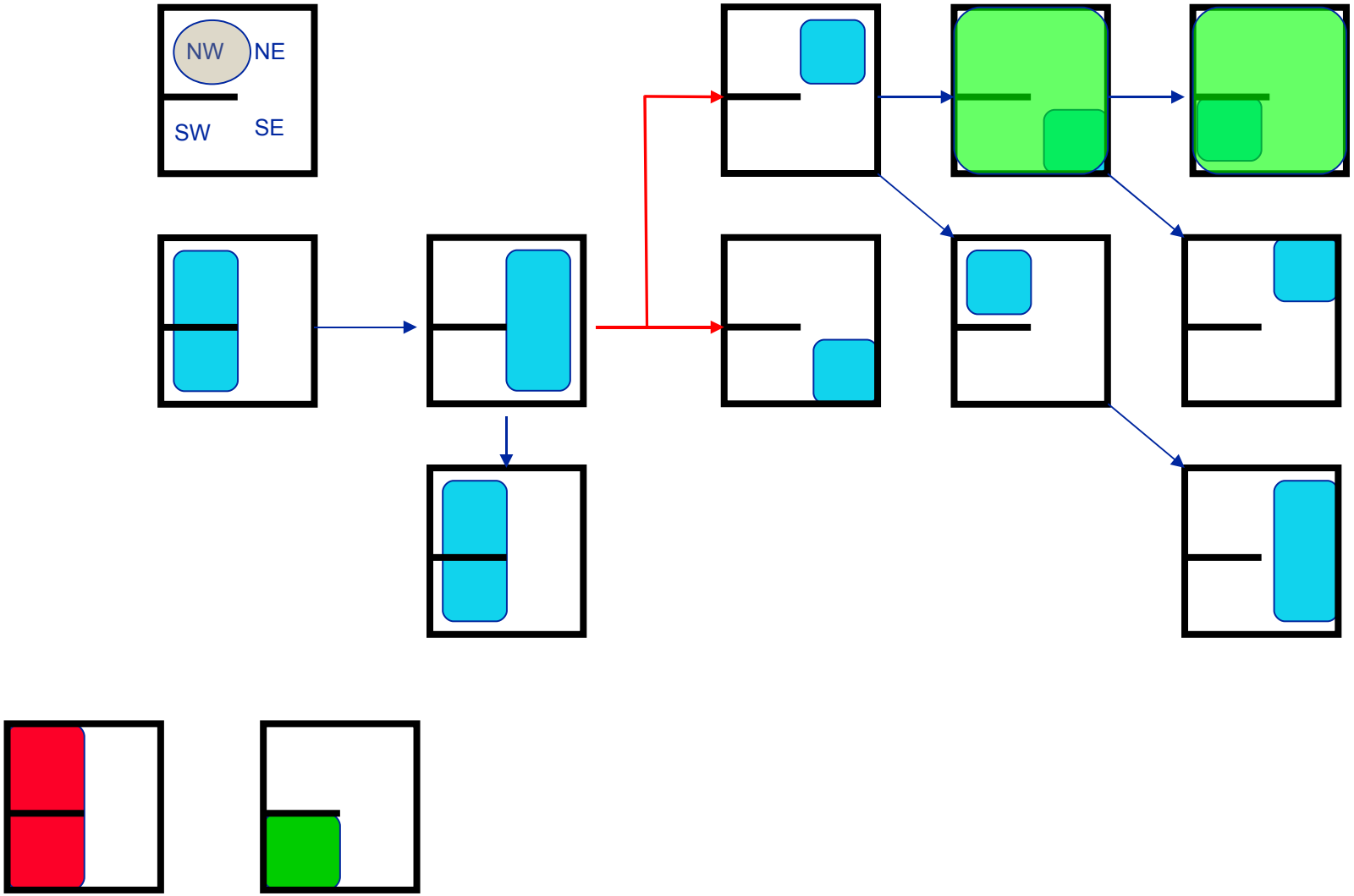


An example

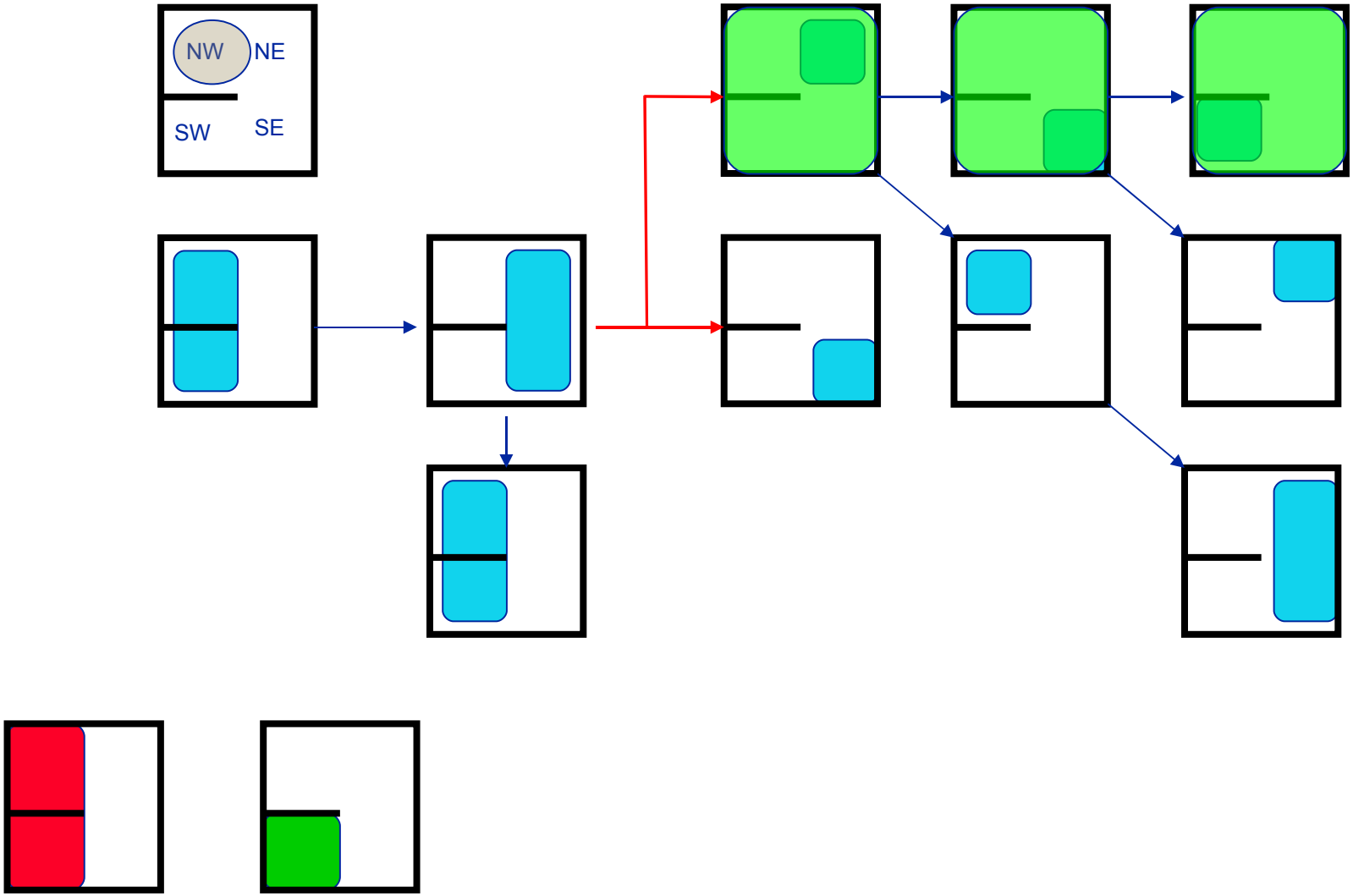




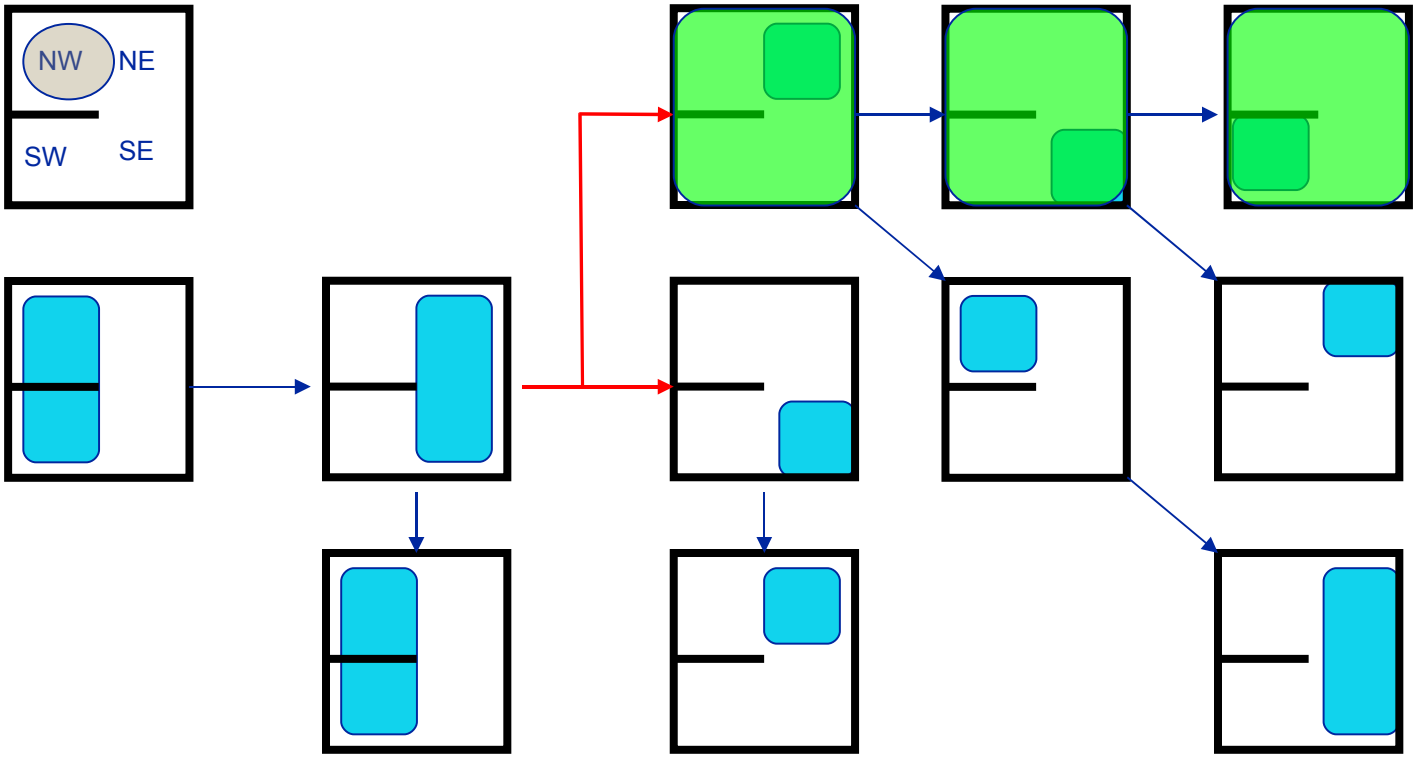
An example

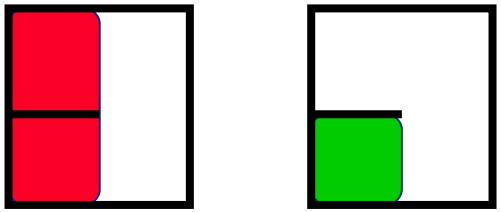


An example

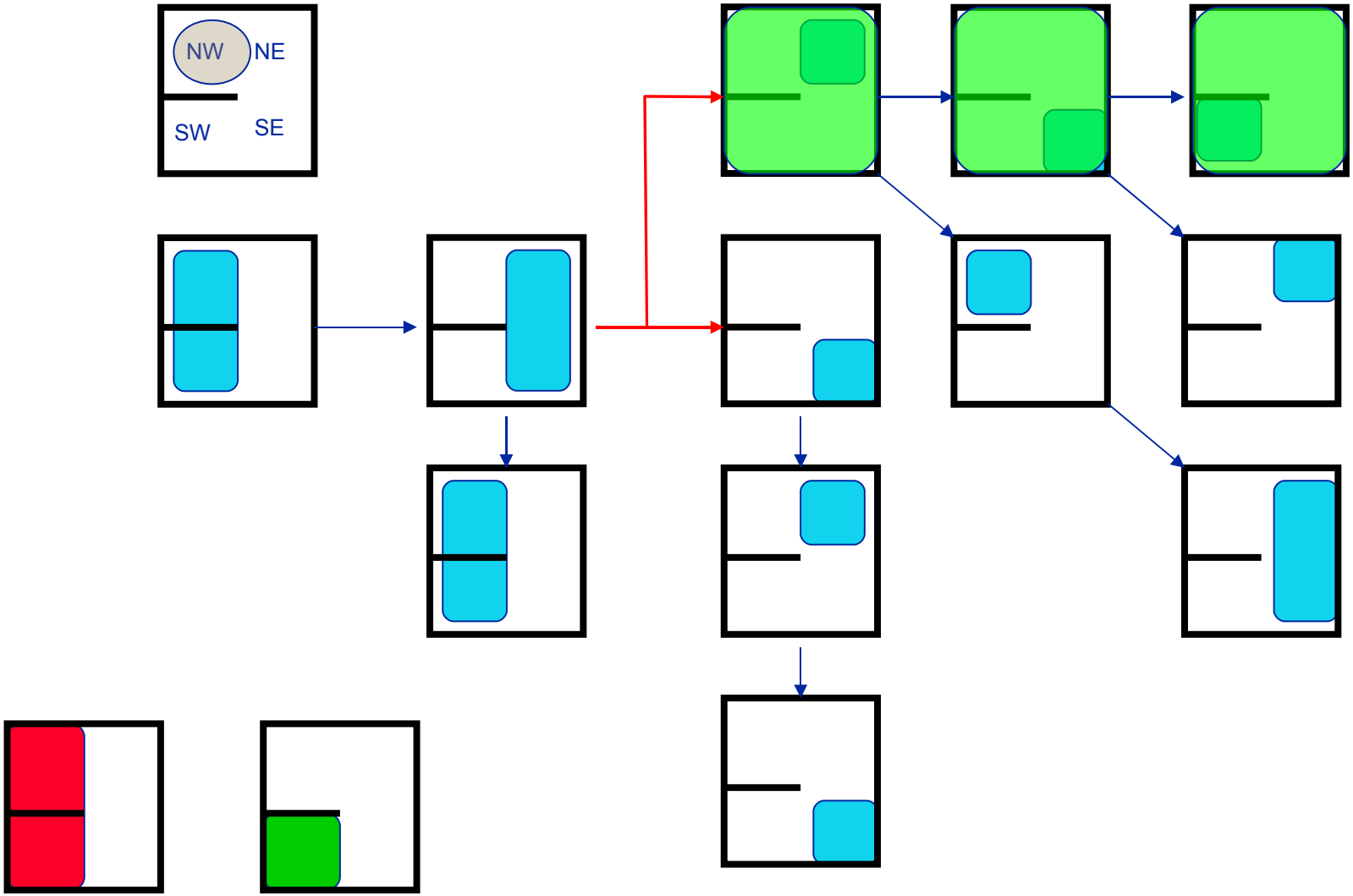


An example

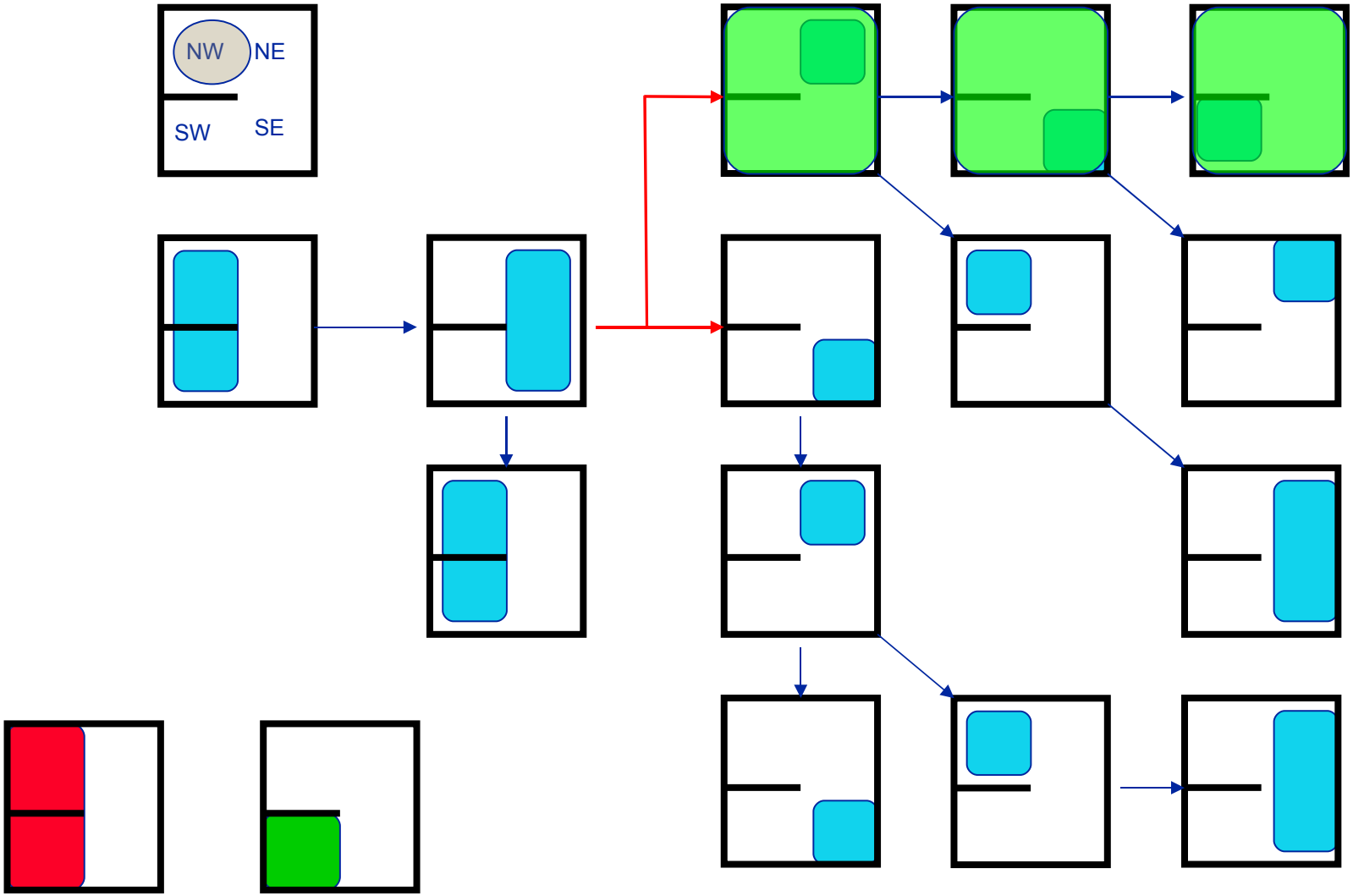




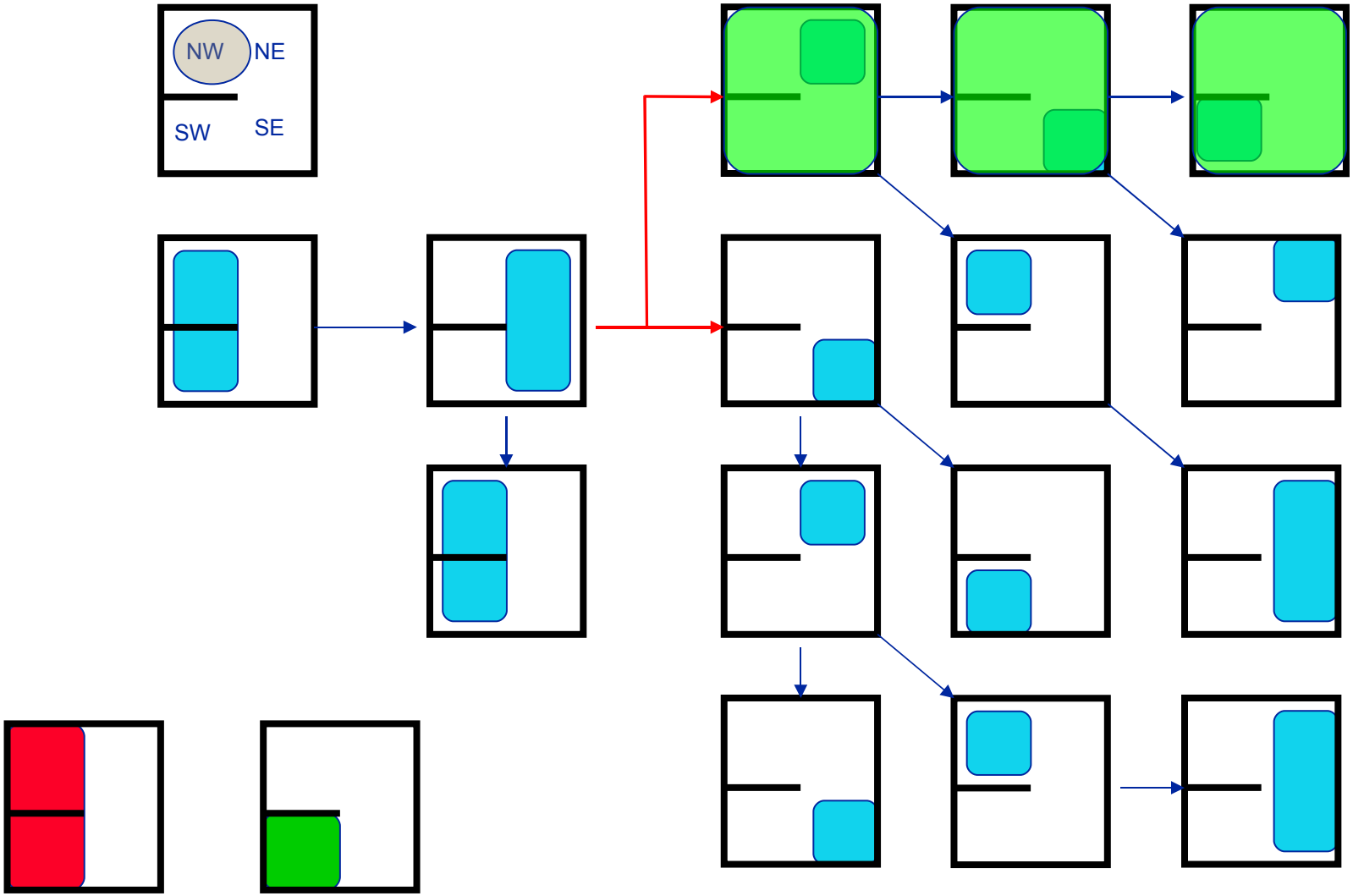
An example



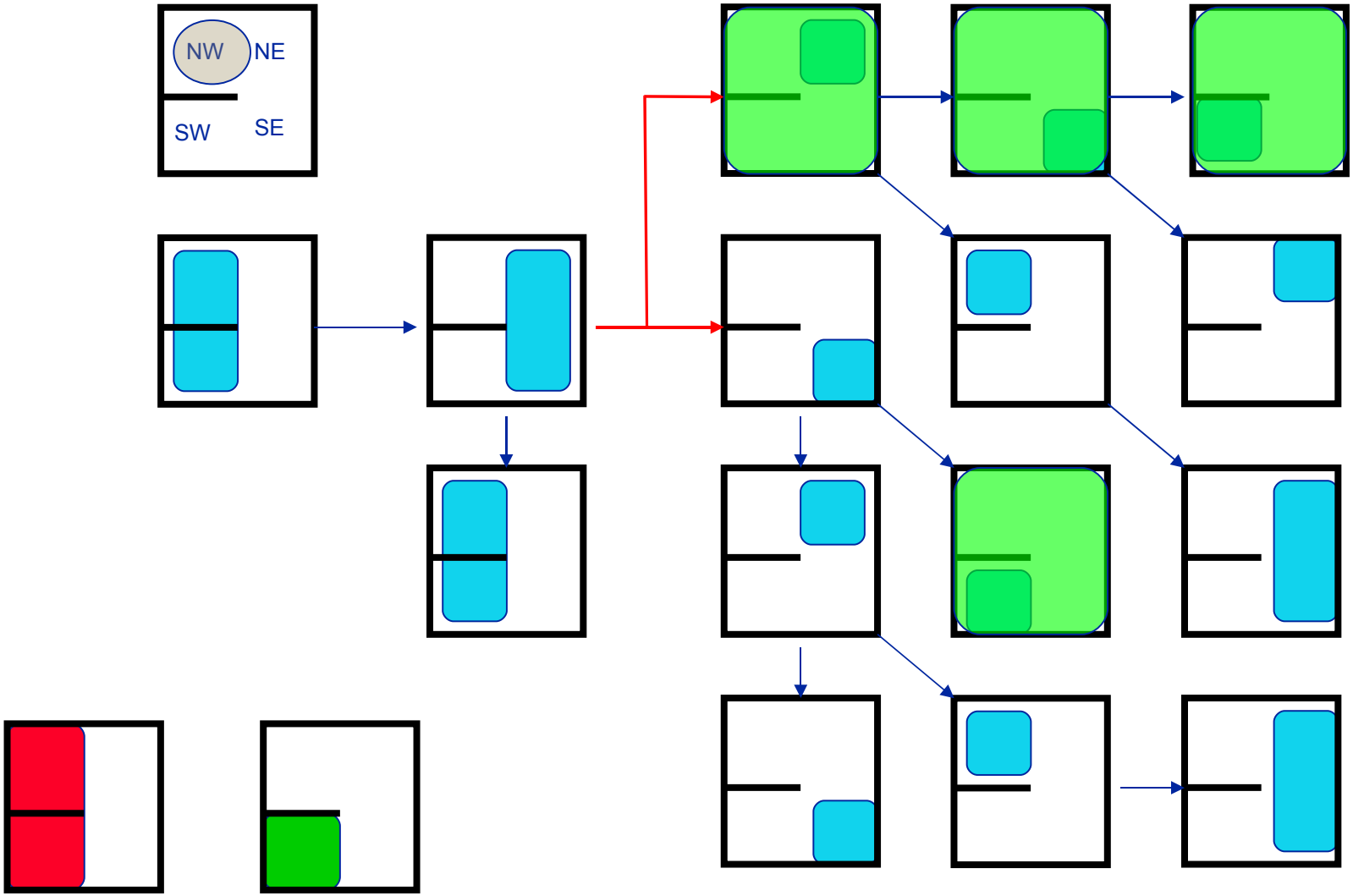
An example



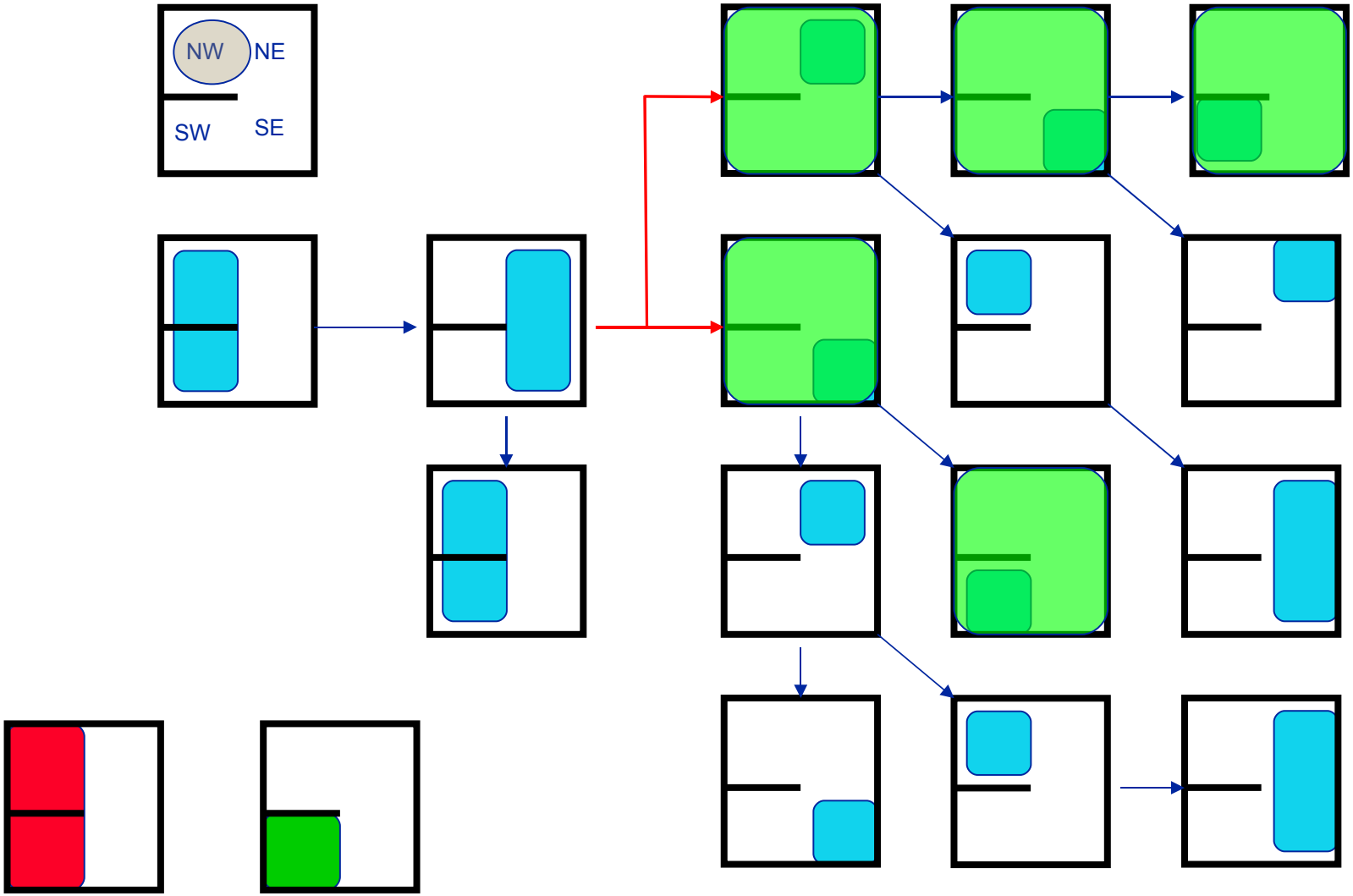
An example



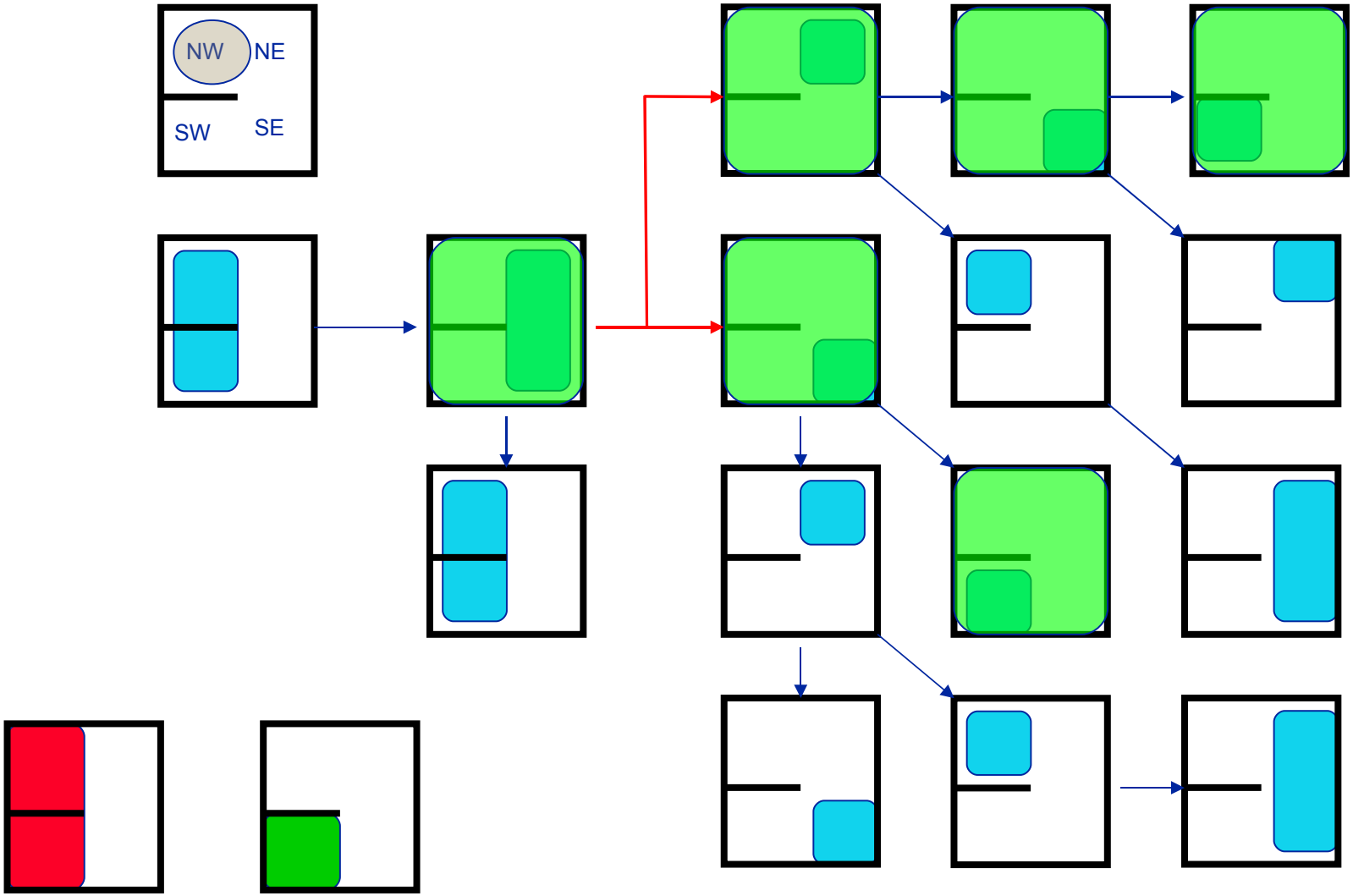
An example



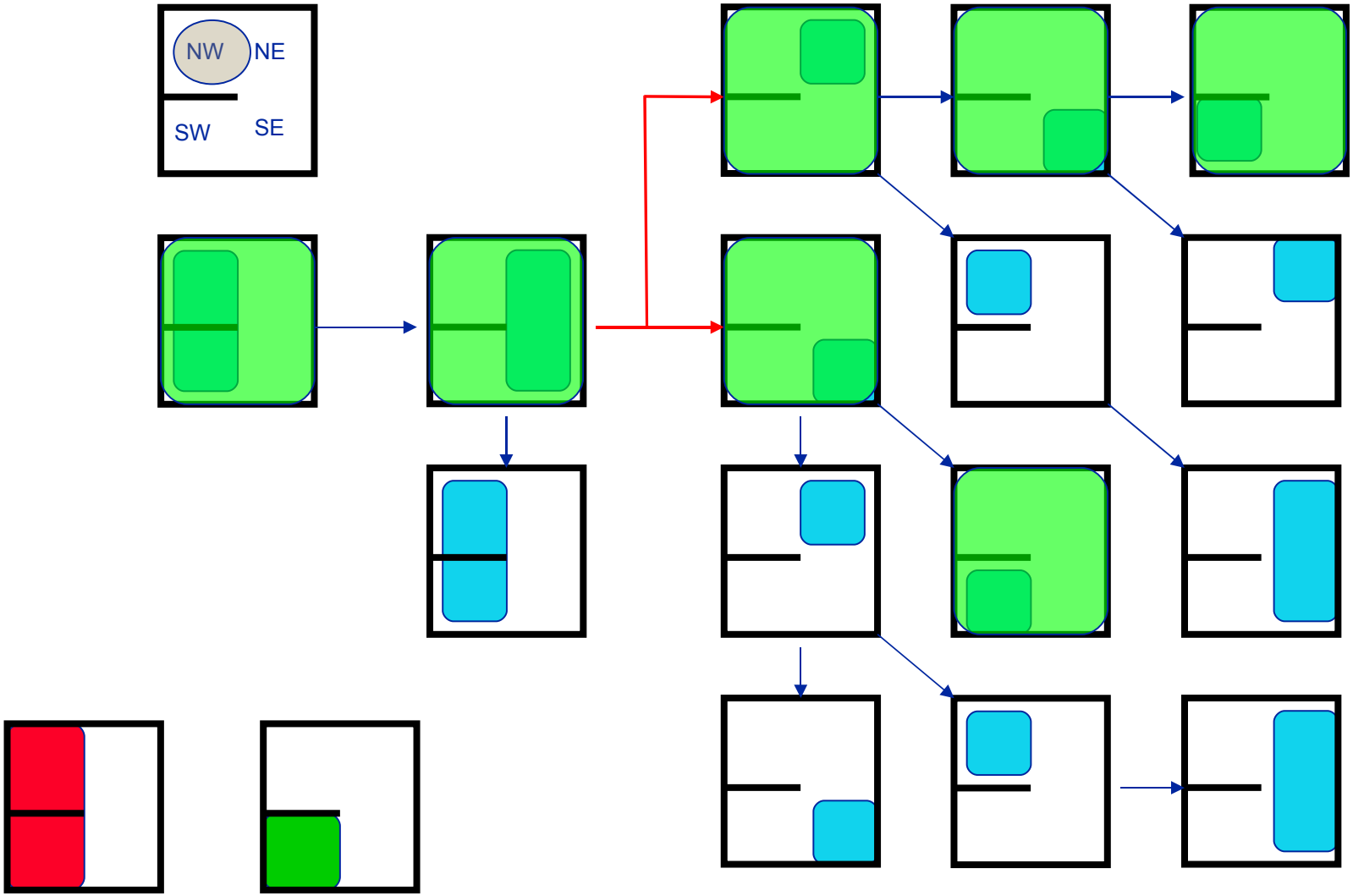
An example



An example



An example



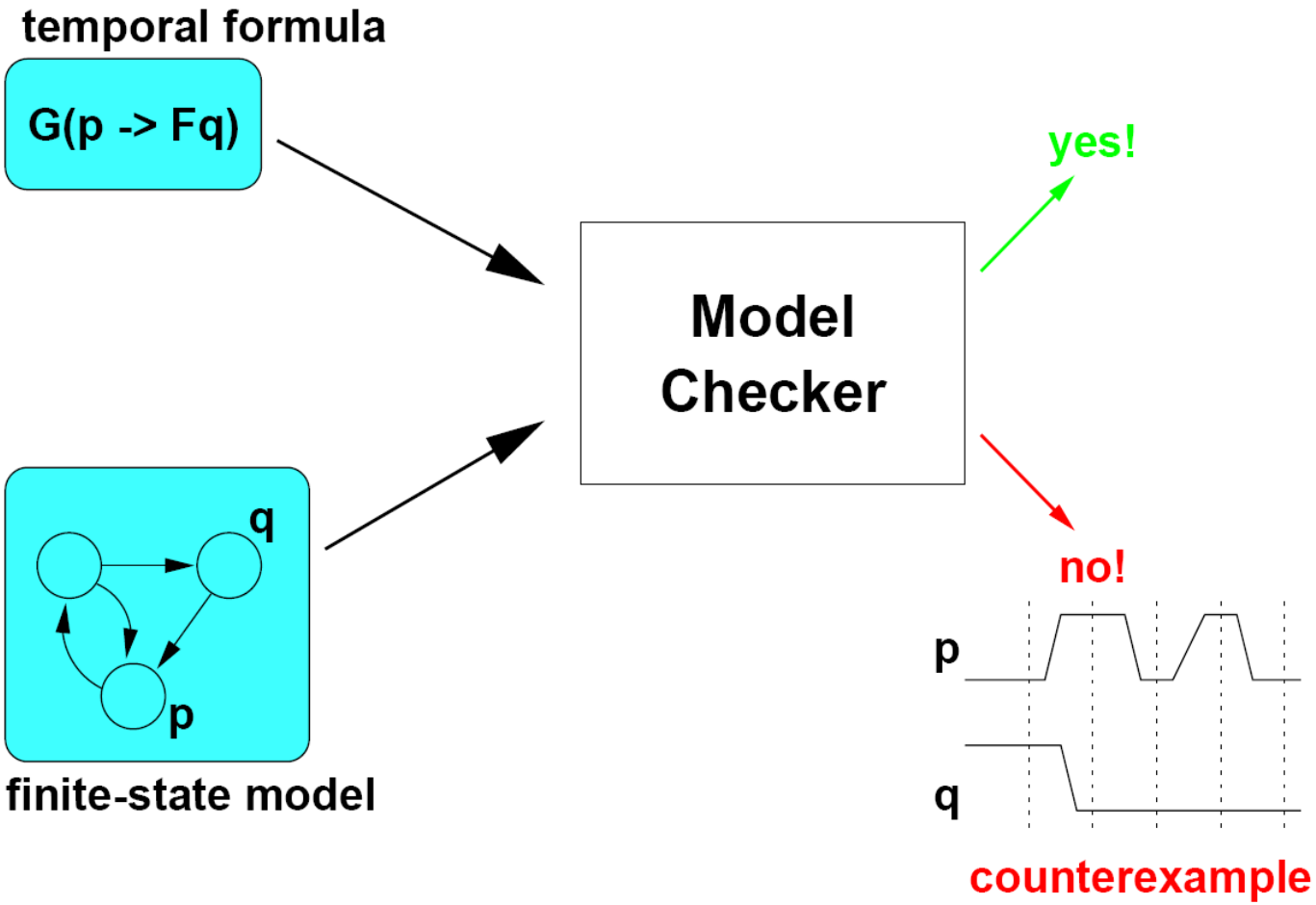
Fault Detection, Identification, Recovery

- While plan executes, performed actions and observed events are recorded
  - $o_0;a_1;o_1;\ \ldots;\ a_k;\ o_k$
- Bounded History Window over Observations and Actions
  - select limited horizon of observation k
  - $X(s_0,o_0)\ \&\ T(s_0,a_1,s_1)\ \&\ X(s_1,o_1)\ \&\ \ldots\ \&\ T(s_{k-1},a_k,s_k)\ \&\ X(s_k,o_k)$
  - defines set of system state sequences compatible with observations and actions
- Construction of a monitor for fault variables
  - $OccF_0$ iff F_0
  - $OccF_i$ iff F_i v $OccF_{i-1}$
- Cross-product with the Model of the controlled plant

- Simulation of the History Window on the cross-product model
  - Accumulate reachable states of the cross-product
  - Project on fault monitor variables
  - Analyze the resulting set to extract the possible faults
  - For multiple faults, consider the one with highest probability

The Role of Symbolic Techniques

Model Checking



Symbolic Representation

- ***Symbolic*** Model Checking
  - use logic-based data structures (e.g. BDD, SAT) for the representation and exploration of model
  - may avoid blow-up due to enumeration
- In particular, Kripke structures are implicitly generated by a language:
  - variable based representation
  - synchronous/asynchronous composition
- Model checking is linear in the size of the structure…
- but the structure is exponential in the number of components

Symbolic Representation

- Vector of state variables **V**
  - e.g. robot in 4x4 square
    - » x : {0, 1, 2, 3}, y : {0, 1, 2, 3}
  - State as assignment to state variables
    - » x = 2, y = 3
- Vector of action variables **A**
  - e.g. robot actions
    - » a : { N, S, E, W}
  - Action as assignment to action variables
    - » a = N
- Vector of next state variables **V'**
  - Transition as assignment to current/next state and action variables
    - » x = 2, y = 3, a = N, x' = 2, y' = 2

Symbolic Representation

- Set-oriented representation
  - sets of states
  - sets of actions
  - sets of state-action pairs (i.e. policies)
  - sets of transitions
- Logic to represent and manipulate characteristic functions of sets
  - conjunction as intersection
  - union as disjunction
  - complementation as negation
  - existential quantification as projection
  - …
- Basic advantage: characteristic function may be much much more compact than represented set
  - x = 0 represents { 0 } and { 00, 01 } and { 000, 001, 010, 011 }

Symbolic Technologies

- Traditionally, based on Binary Decision Diagrams
  - canonical form for propositional logic
  - dependence on the ordering
  - fix point algorithms
  - QBF primitives
- More recently, SAT based model checking
  - Bounded Model Checking
    - » extension to LTL for SAT-plan approach
  - Invariant checking for inductive reasoning
- In this study, BDD-based technologies

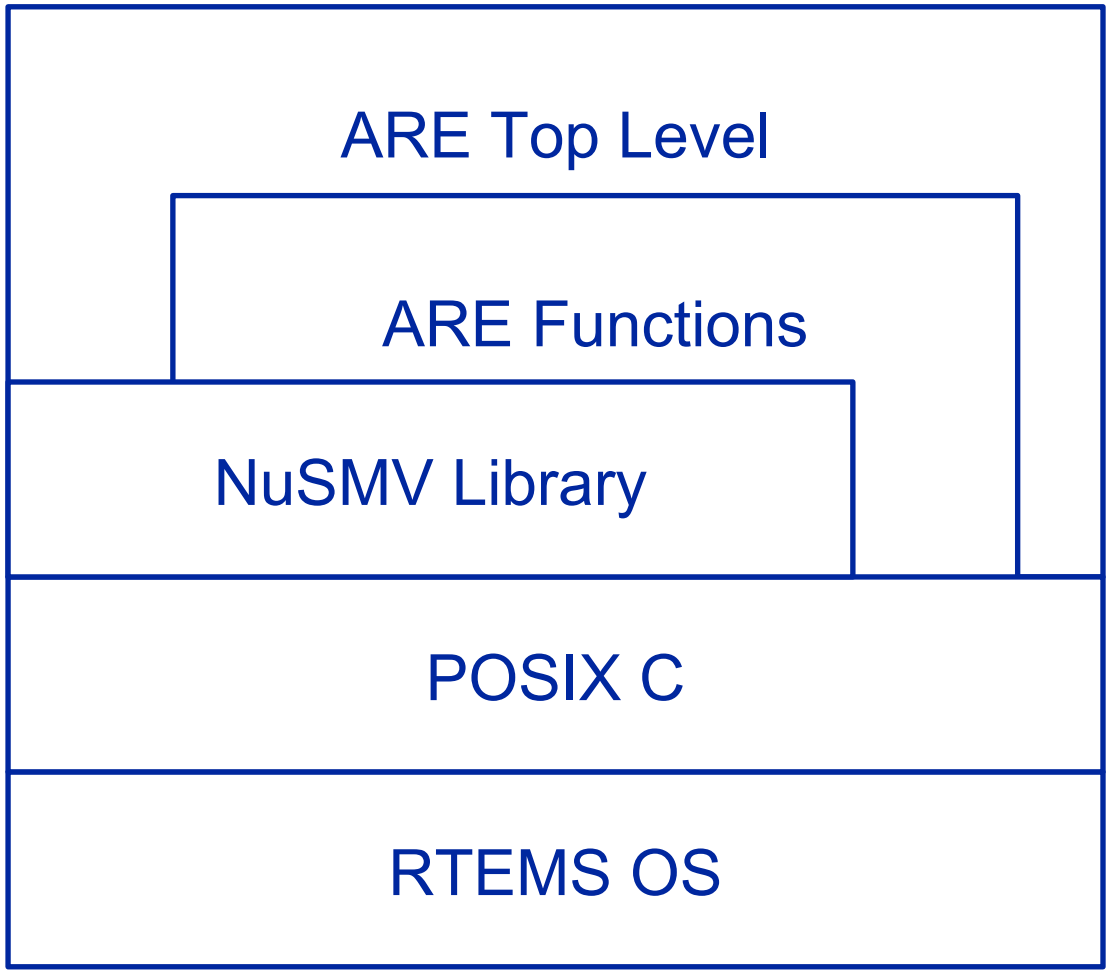
Binary Decision Diagrams

- The BDD package provides
  - boolean operations
  - universal and existential quantification (QBF)
  - caching and memoizing
- Used to represent
  - accumulated states
  - belief states
  - state distances for computation of heuristics
  - assumptions and plan annotations
- Used to compute
  - equality, containment between belief states
  - set operations between belief states (union, intersection, …)
  - images in reachability analysis
  - belief state progression

The ARE Implementation

Marco Roveri

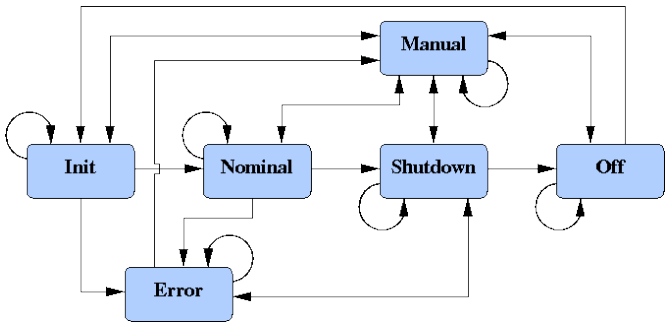
ARE implementation

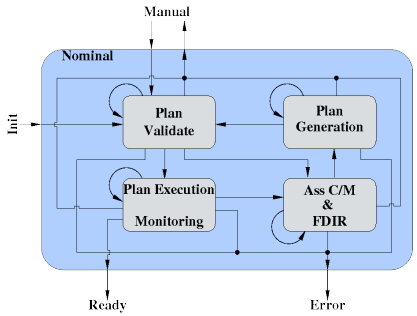


ARE implementation (II)

Command from Ground

ARE FSM Thread





ARE Child Thread

- initialization
- plan validation
- plan generation
- plan execution and monitoring
- FDIR
- Resource Estimation

Command to actuators

Sensing information

Implementation considerations

- No dynamic memory allocation
- Bounded (controlled) recursion
- Child thread is scheduled as a sporadic thread

Implementation considerations (II)

- Development language
  - ISO ANSI C API
  - POSIX 1003.1B API
- Developed code checked for
  - buffer over-runs and under-runs using the DUMA and Valgrind suites
  - use of variables before initialization with Valgrind suite
  - thread synchronization errors with Valgrind
    - » http://duma.sourceforge.net
    - » http://valgrind.org
- NuSMV coding standard used for the whole project

Implementation considerations (III)

- Developed lines of code (no domain dependent code)
  - ARE Top Level:

	# Files	Comments	code
C headers	6	621	733
C sources	3	130	115
SUM	9	751	848

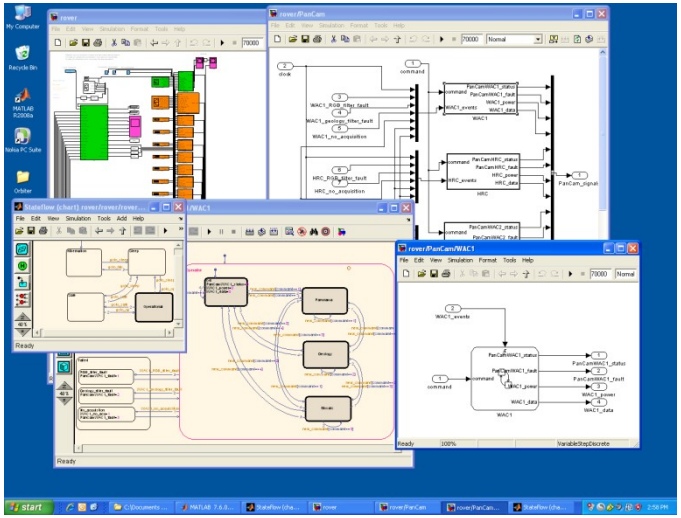
  - ARE Functions:

	# Files	Comments	code
C headers	19	1023	644
C sources	20	4753	5743
SUM	39	5776	6387

Cfr. cloc: http://cloc.sourceforge.net/

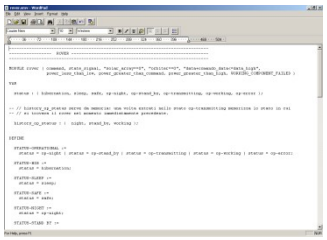
Model Generation Flow

Matlab/Simulink/Stateflow model



NuSMV model generation

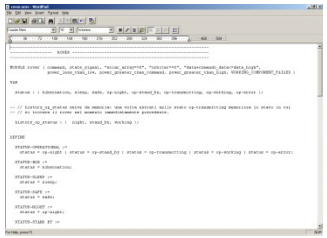
NuSMV model



Model validation with NuSMV

Feedback to the Matlab/Simulink/Stateflow model

Generation of Low Level Resource Function



Model and Resource Functions

	C comments	C code	SMV lines
Rover	296	4669	1854
Small Rover	263	1789	524
Orbiter	173	480	181

Cfr. cloc: http://cloc.sourceforge.net/

- About 310 properties to validate Rover model
- About 50 properties to validate Orbiter model

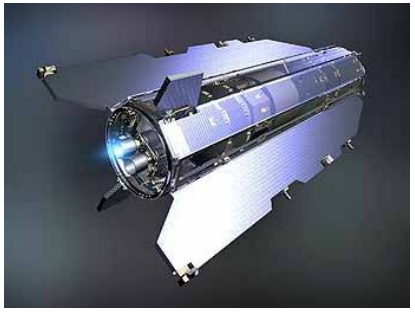
The ARE Characterization

Marco Roveri

Evaluation approach

- Case studies
  - Planetary rover
    - » Model taken from another running project developed in ThalesAlenia space

  - Orbiting spacecraft
    - » ThalesAlenia space in house simple model



- Characterization
  - Functional – on desktop PC
  - Embedded – on platform

Functional characterization

- Rover
  - 17 subcomponents (92 commands)
    - » raman (21), spds (12), micromega (3), marsxrd (2), urey (3), moma (4), 2 pancam WAC1 (4), pancam HRC (3), sas (4), locomotion (2), mima (6), wisdom (7), clupi (3), mimosii (3), mamis (4), rat (2), mast (3), arm (2)
- Small rover
  - 3 subcomponents (19)
    - » spds (12), urey (3), sas (4)
- Orbiter
  - 3 subcomponents (9)
    - » imager (3), altimeter (3), antenna (3)

Functional characterization

- For each case study
  - Size of the models
    - » state variables
    - » action variables
    - » observations
    - » total boolean variables
  - Size of state space
    - » potential states
    - » reachable states
    - » size of BDD
    - » diameter (max distance from initial states)
  - Planning
    - » Explored search space
    - » Max depth of plan

Functional characterization

	# state vars	# action vars	# obs	# bool vars	state space	reachable state space	reach. BDD size	diameter
Rover	116	2	74	423	1.00361e+61 (2^202)	3.44562e+58 (2^194)	333	61
Small rover	40	2	20	147	2.90536e+20 (2^67)	2.33589e+19 (2^64)	103	31
Orbiter	16	3	5	77	4.22786e+09 (2^31)	1.6535e+07 (2^23)	106	33

Functional characterization

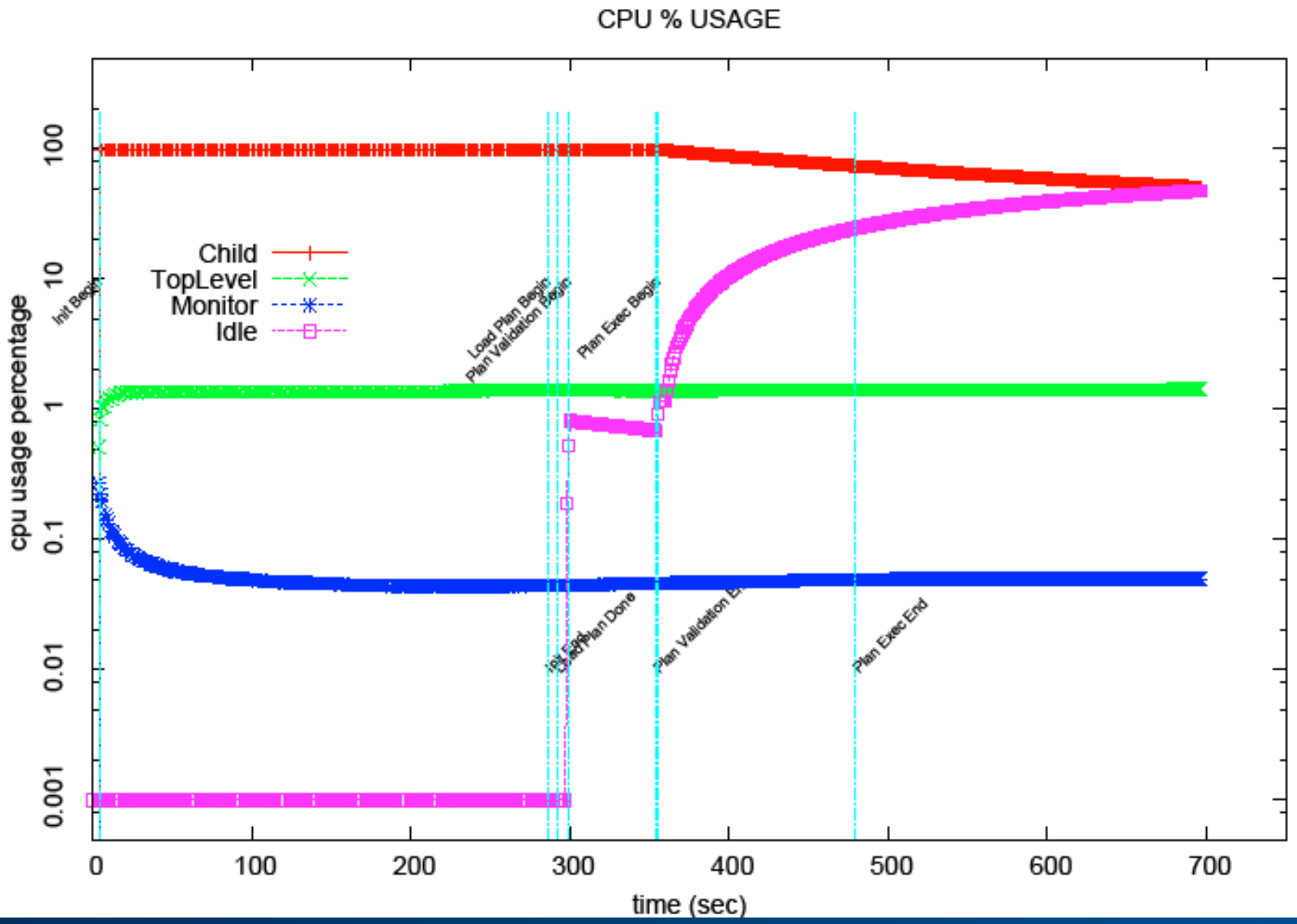
- Plan generation

	# generated nodes	# processed nodes	Plan depth
Rover	323	16	15
Small rover	70	16	15
Orbiter	17	3	2

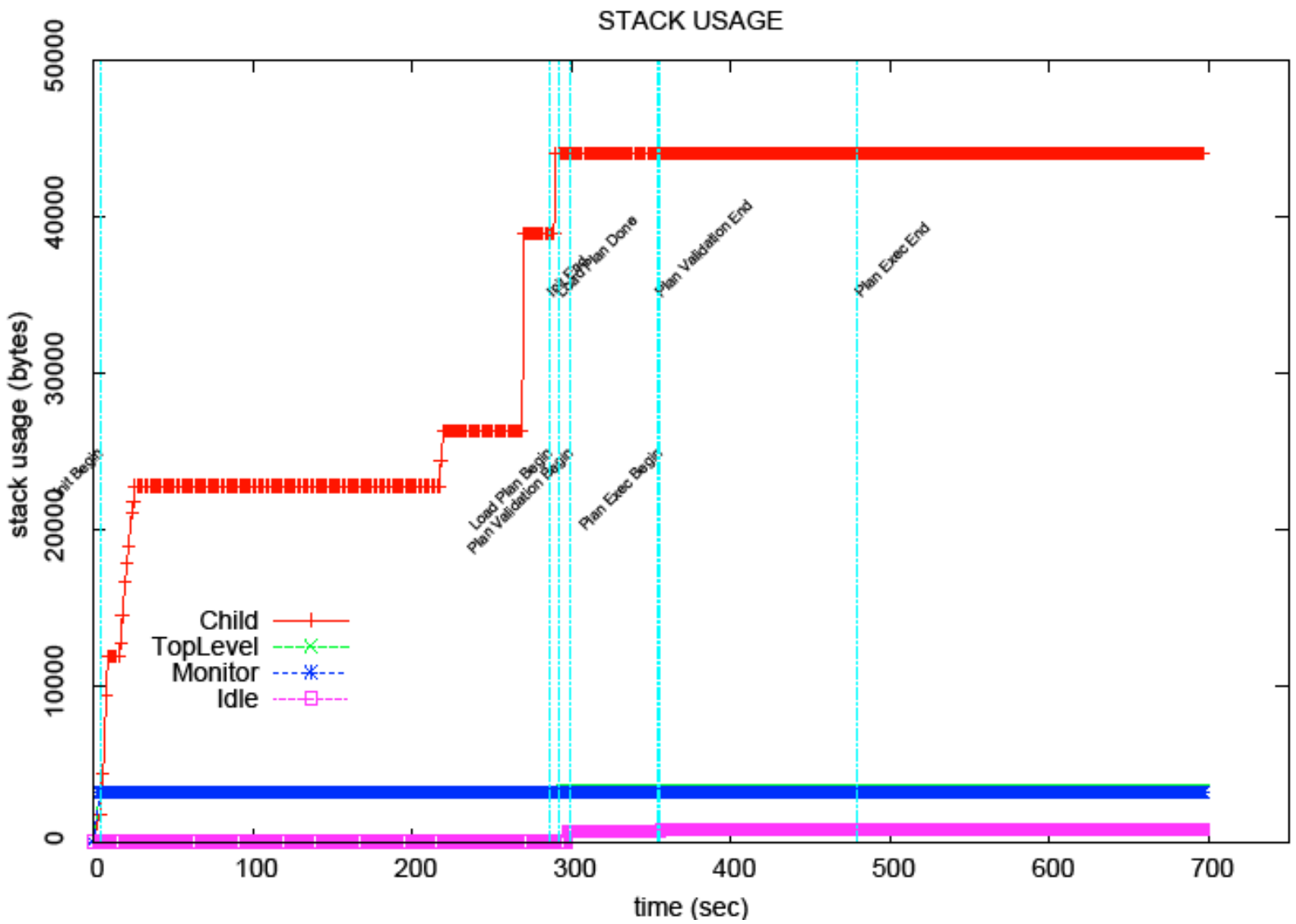
Platform Characterization

- Different scenarios
- Different plots CPU/Stack/Memory
  - Memory characterization to estimate memory requirements
- Different configuration of the rover
  - To characterize the impact of the model
- Different target space architecture
  - Sparc ERC32, Sparc LEON3
  - To evaluate the performance in current and future target CPU architectures

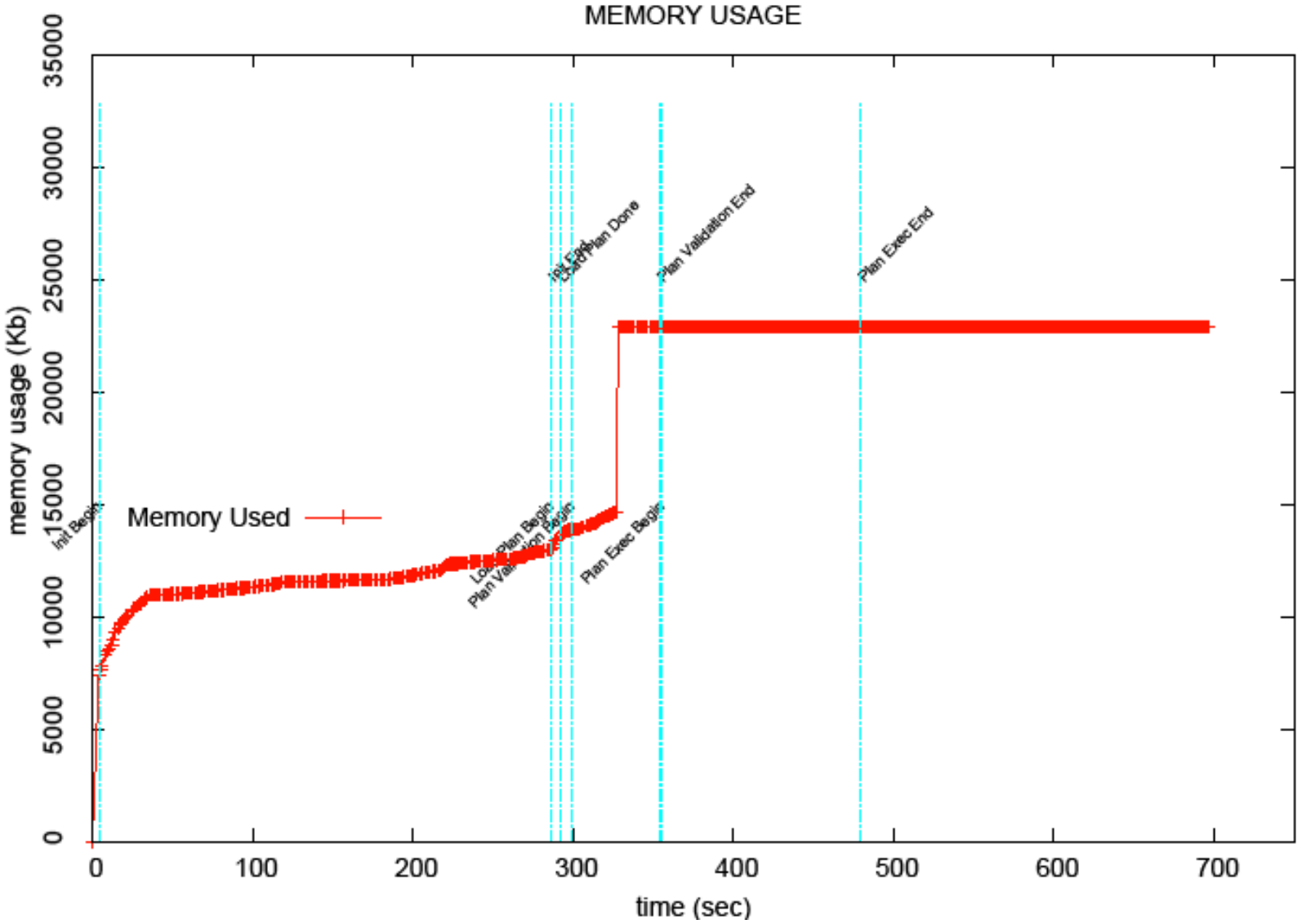
CPU Usage: rover (ERC32)



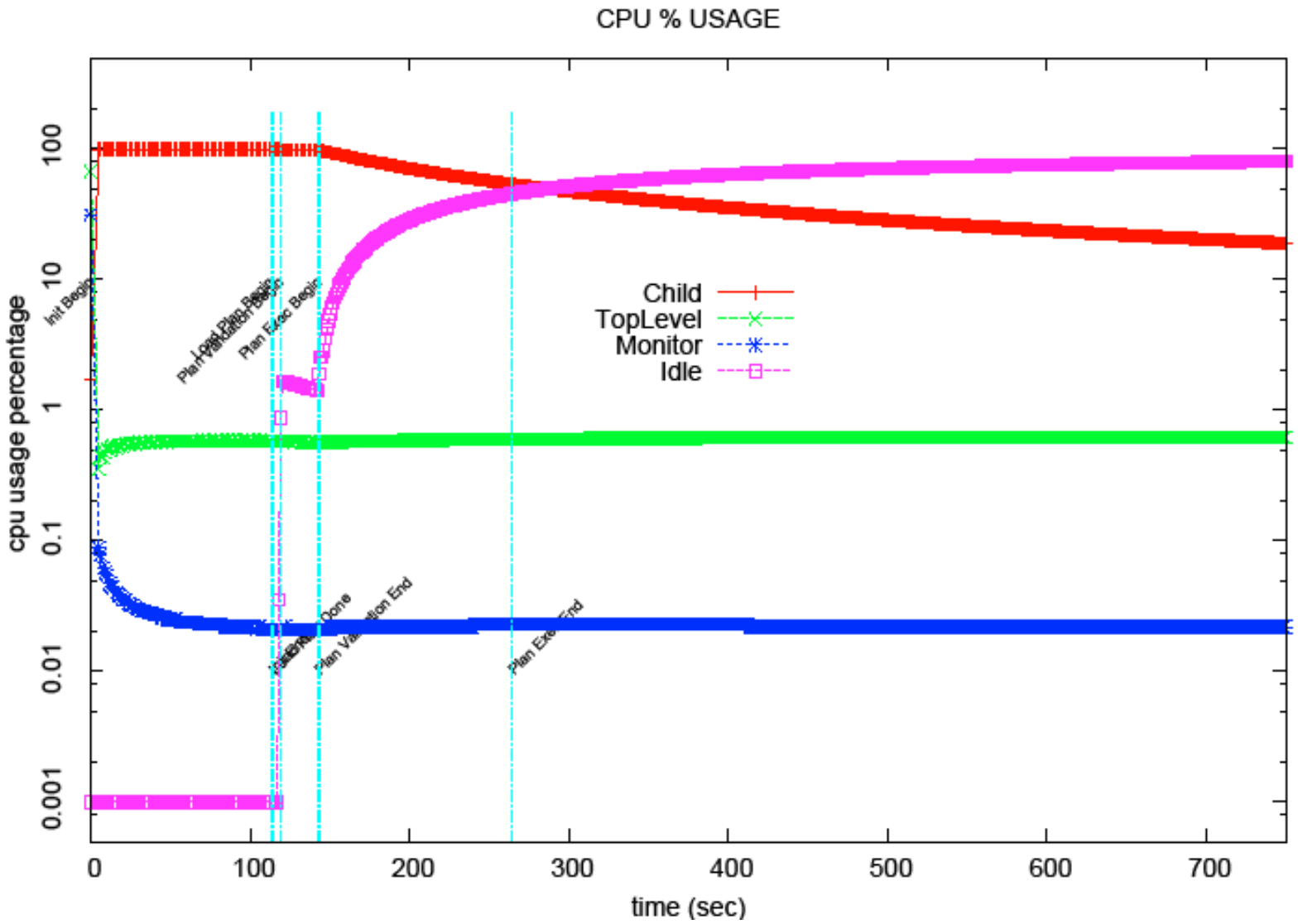
Stack usage: rover (ERC32)



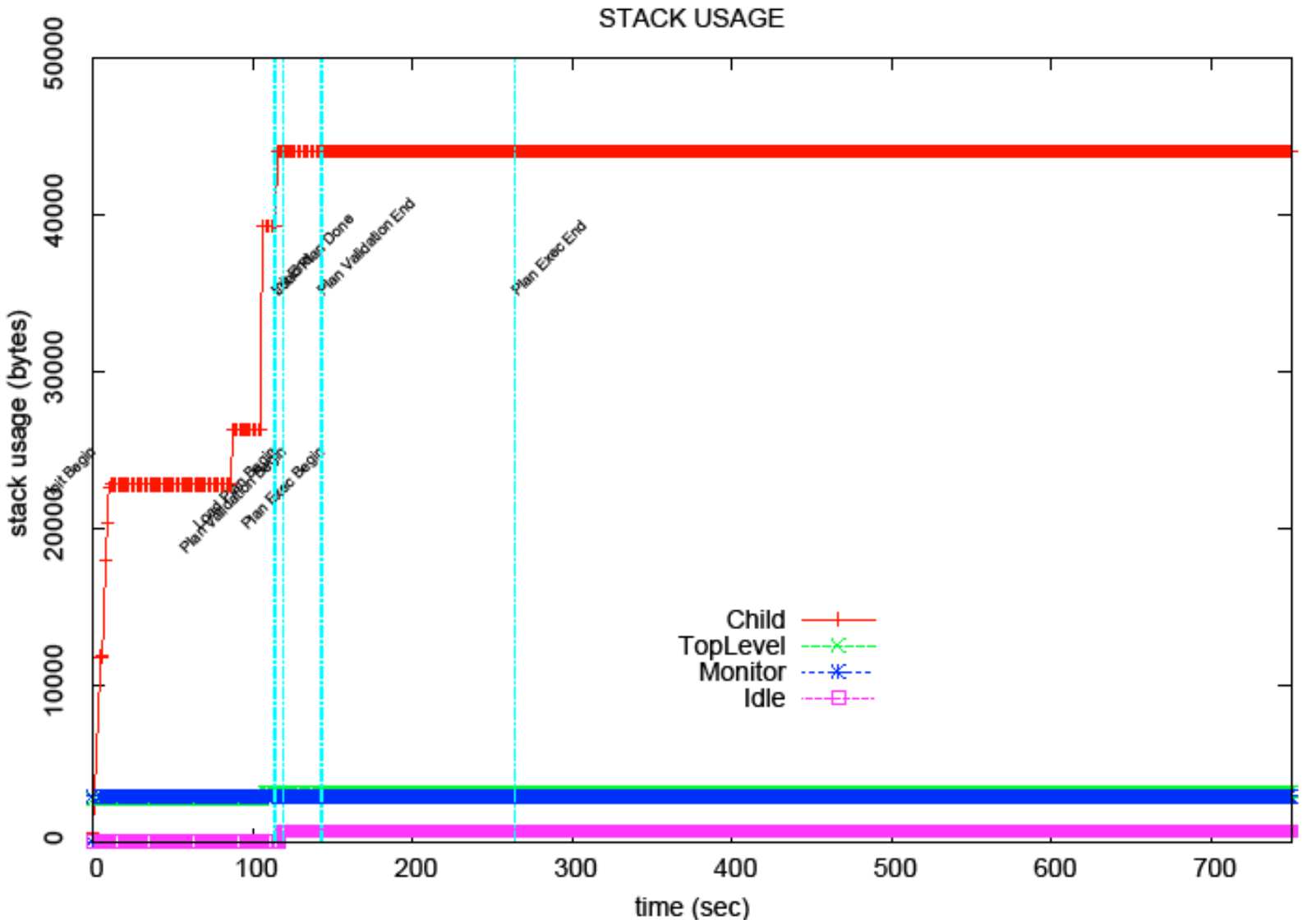
Memory usage: rover (ERC32)



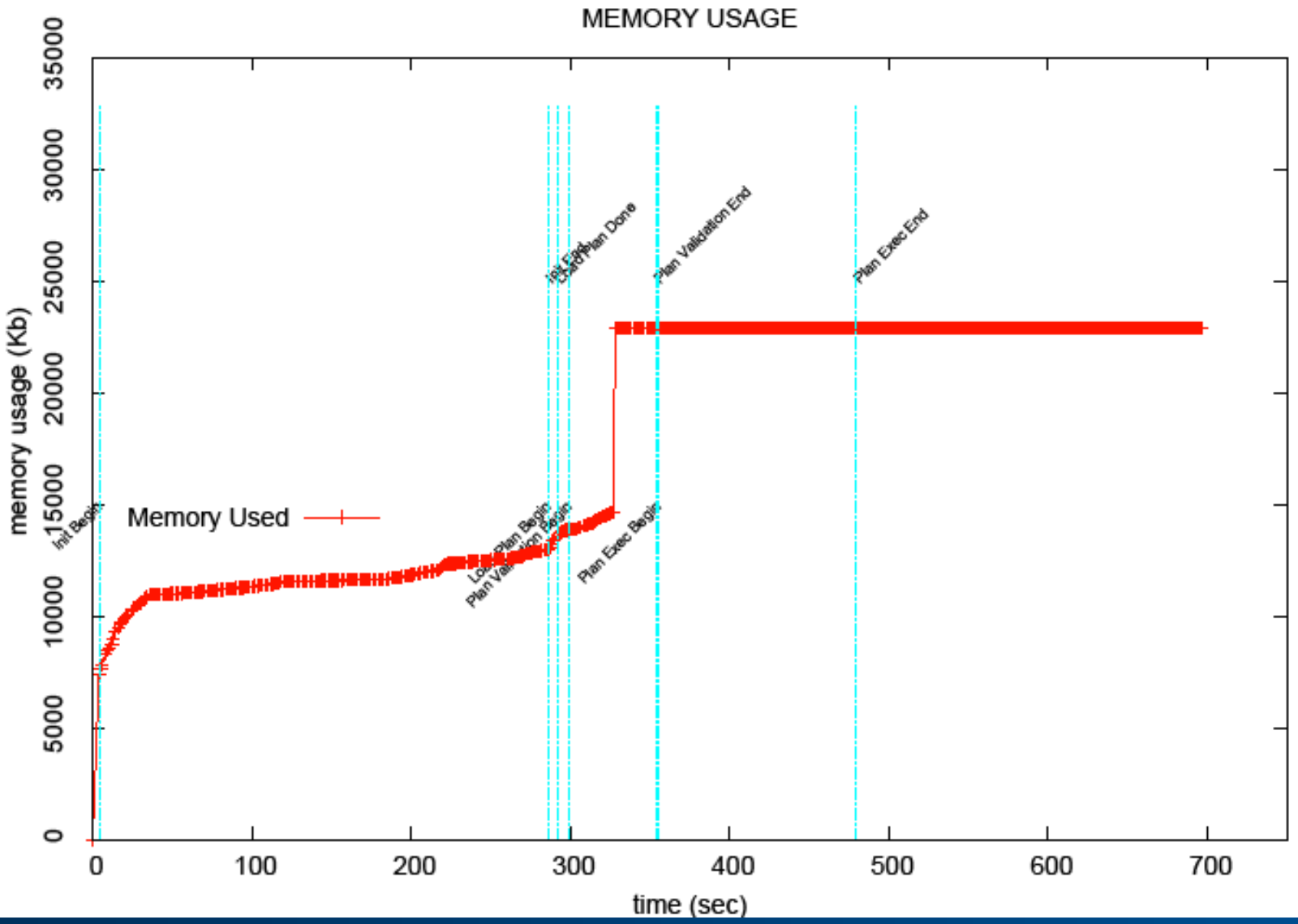
CPU Usage: rover (LEON3)



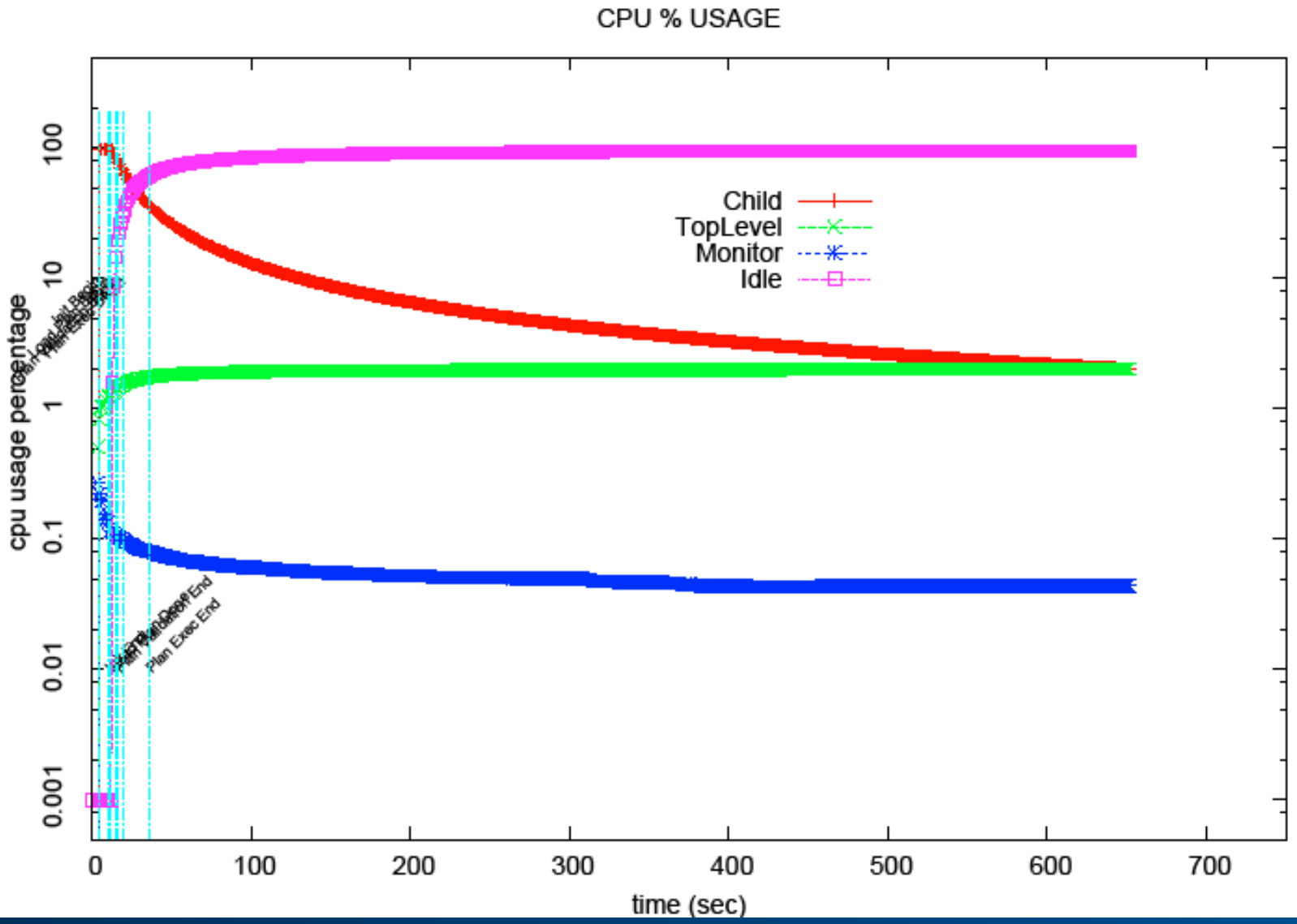
Stack usage: rover (LEON3)



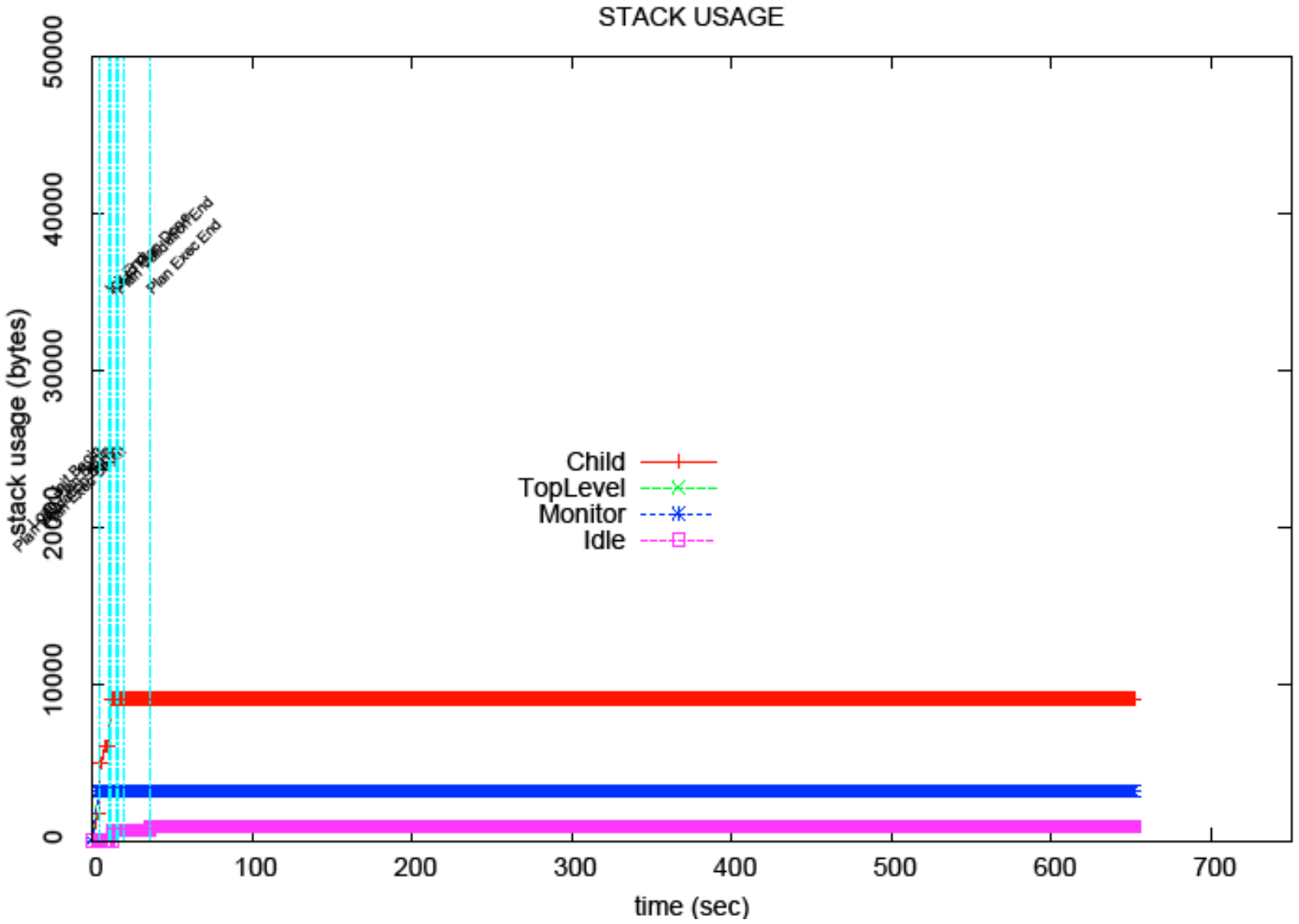
Memory usage: rover (LEON3)



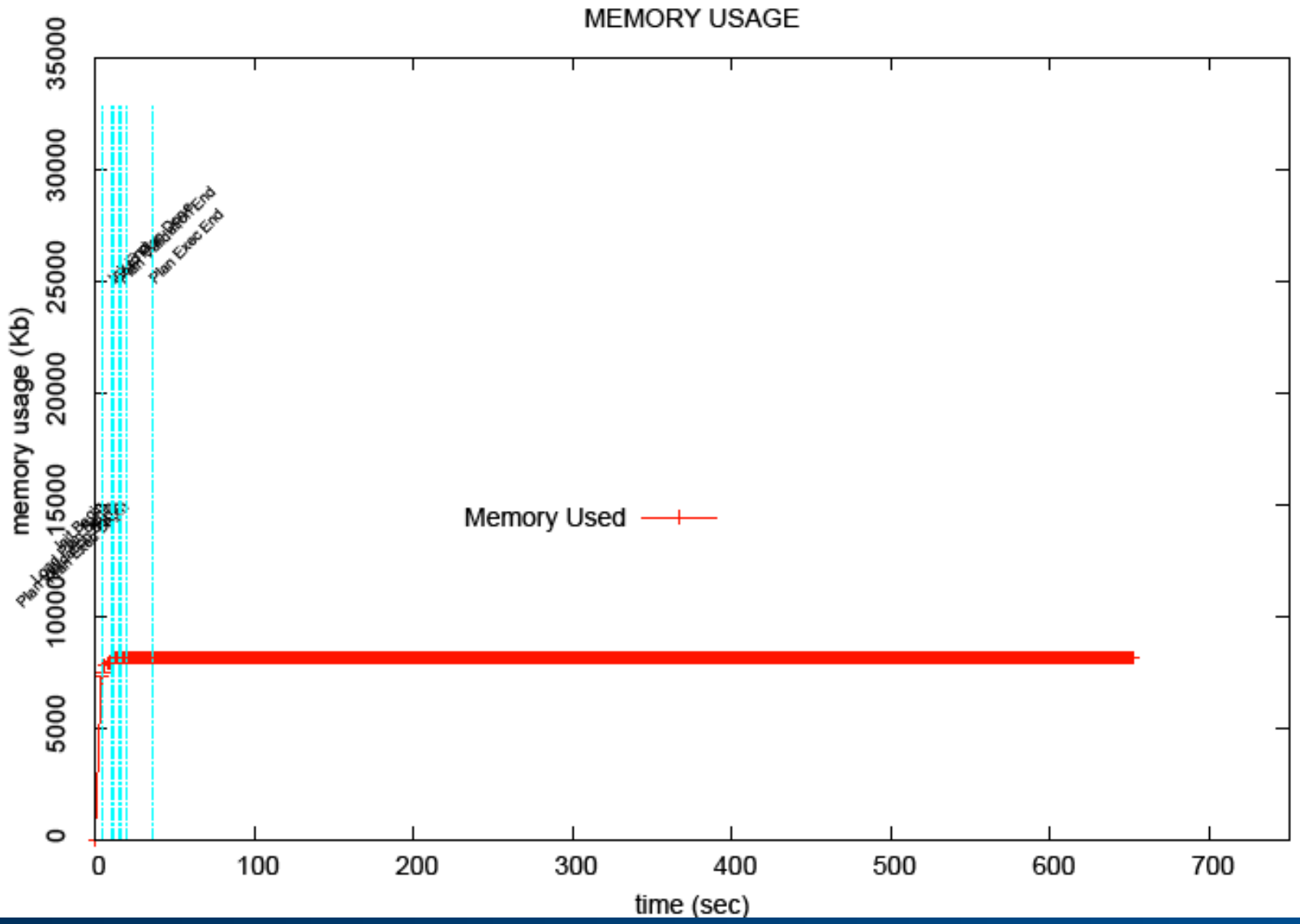
CPU usage: orbiter (ERC32)



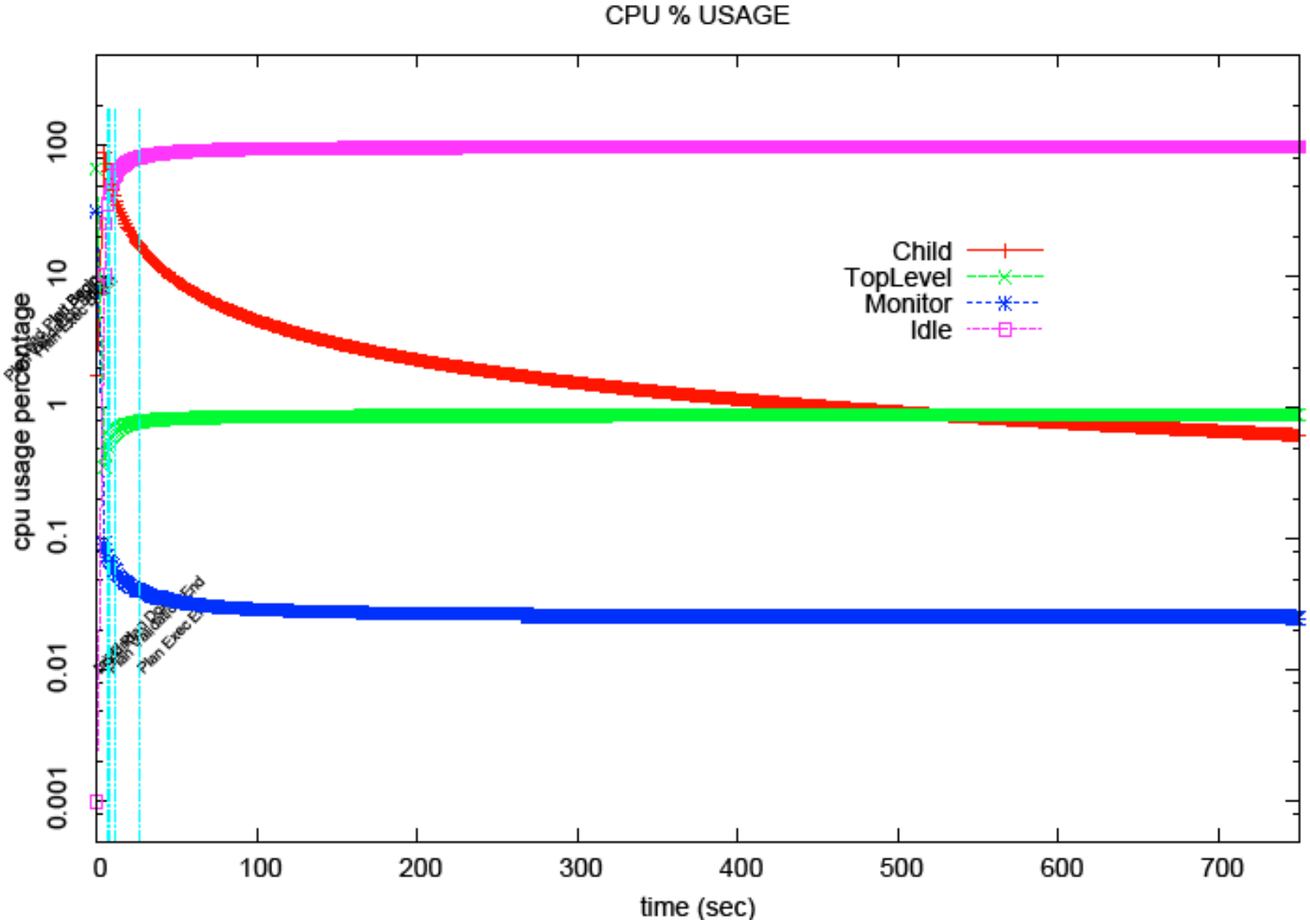
Stack usage: orbiter (ERC32)



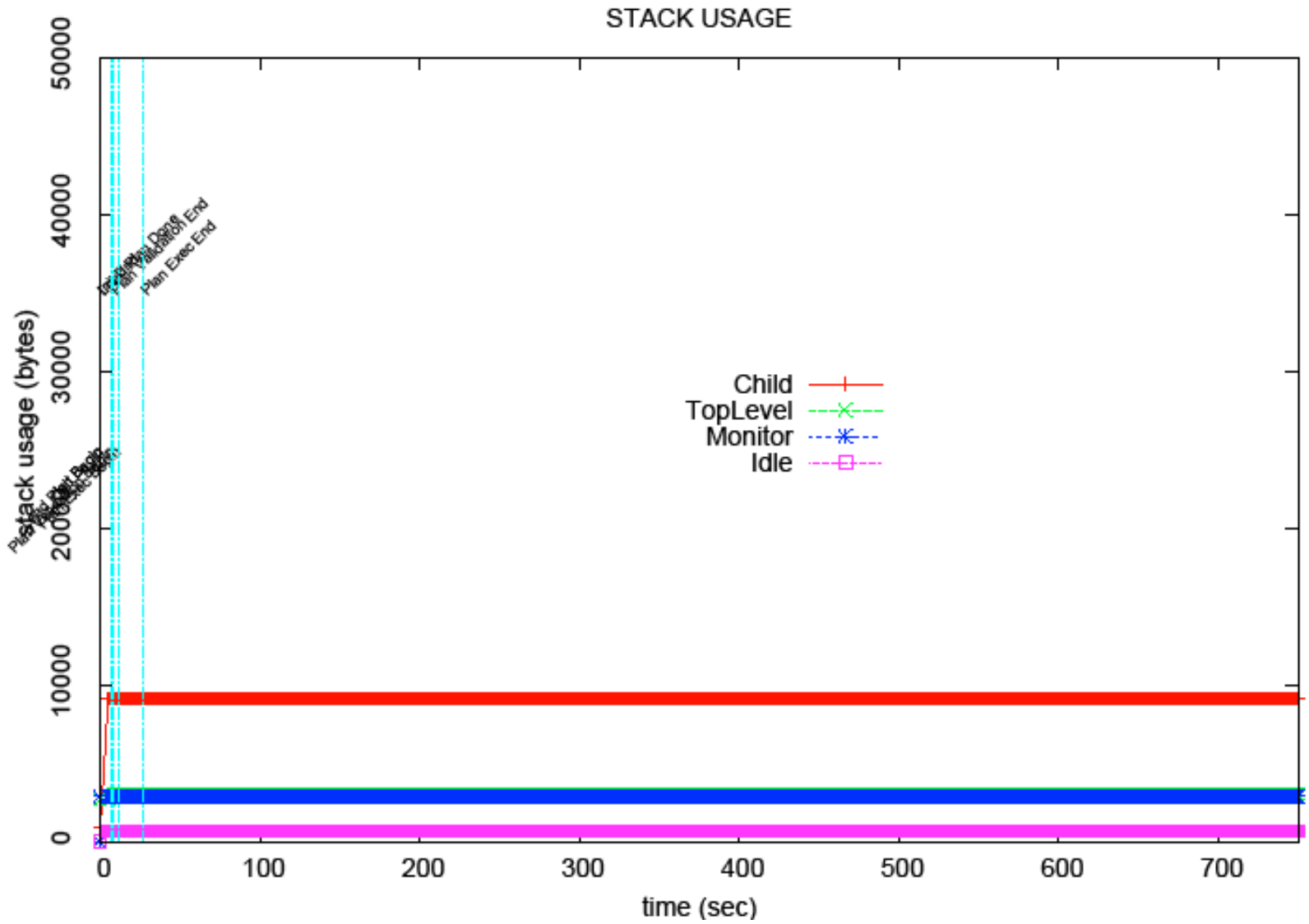
Memory usage: orbiter (ERC32)



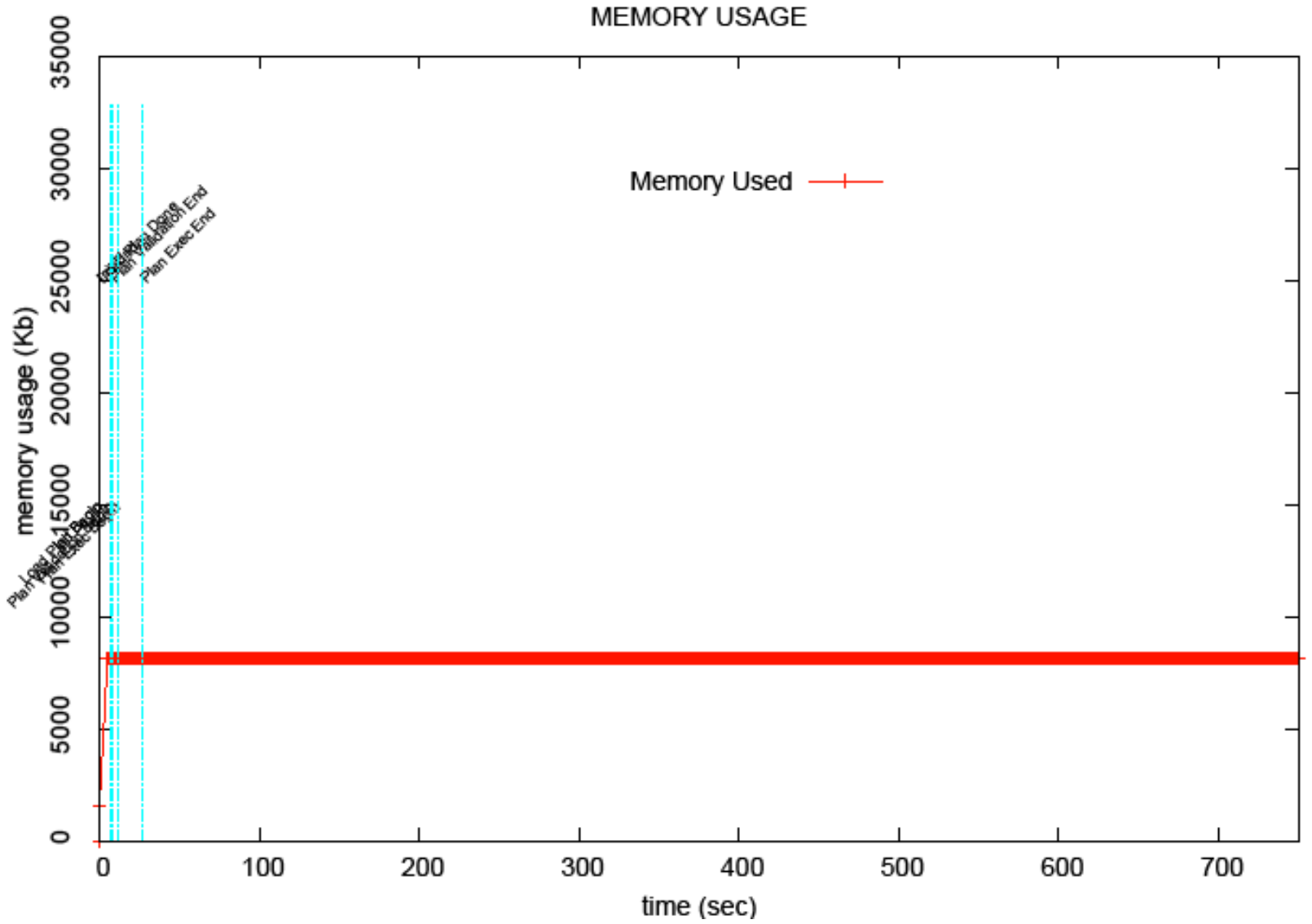
CPU usage: orbiter (LEON3)



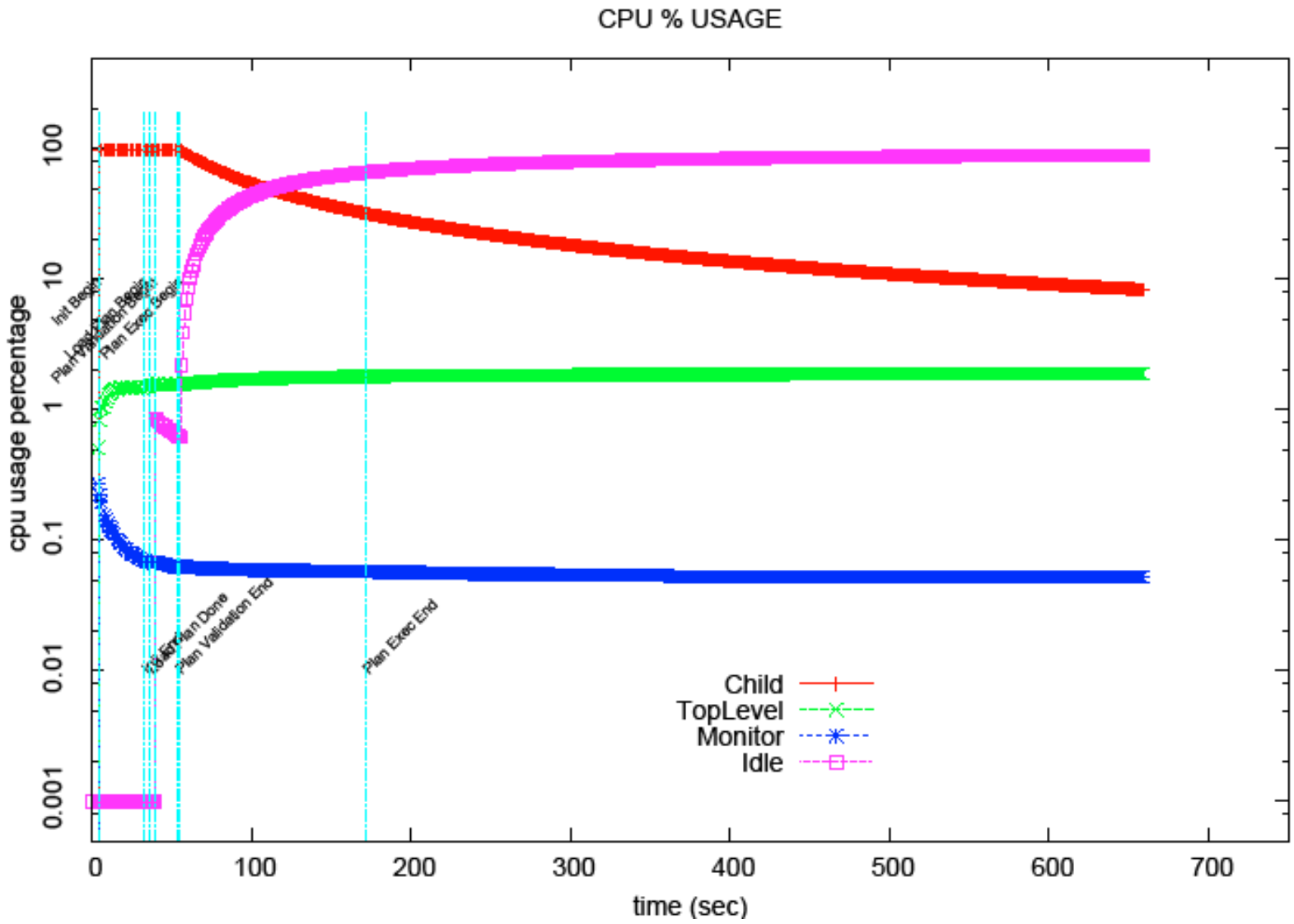
Stack usage: orbiter (LEON3)



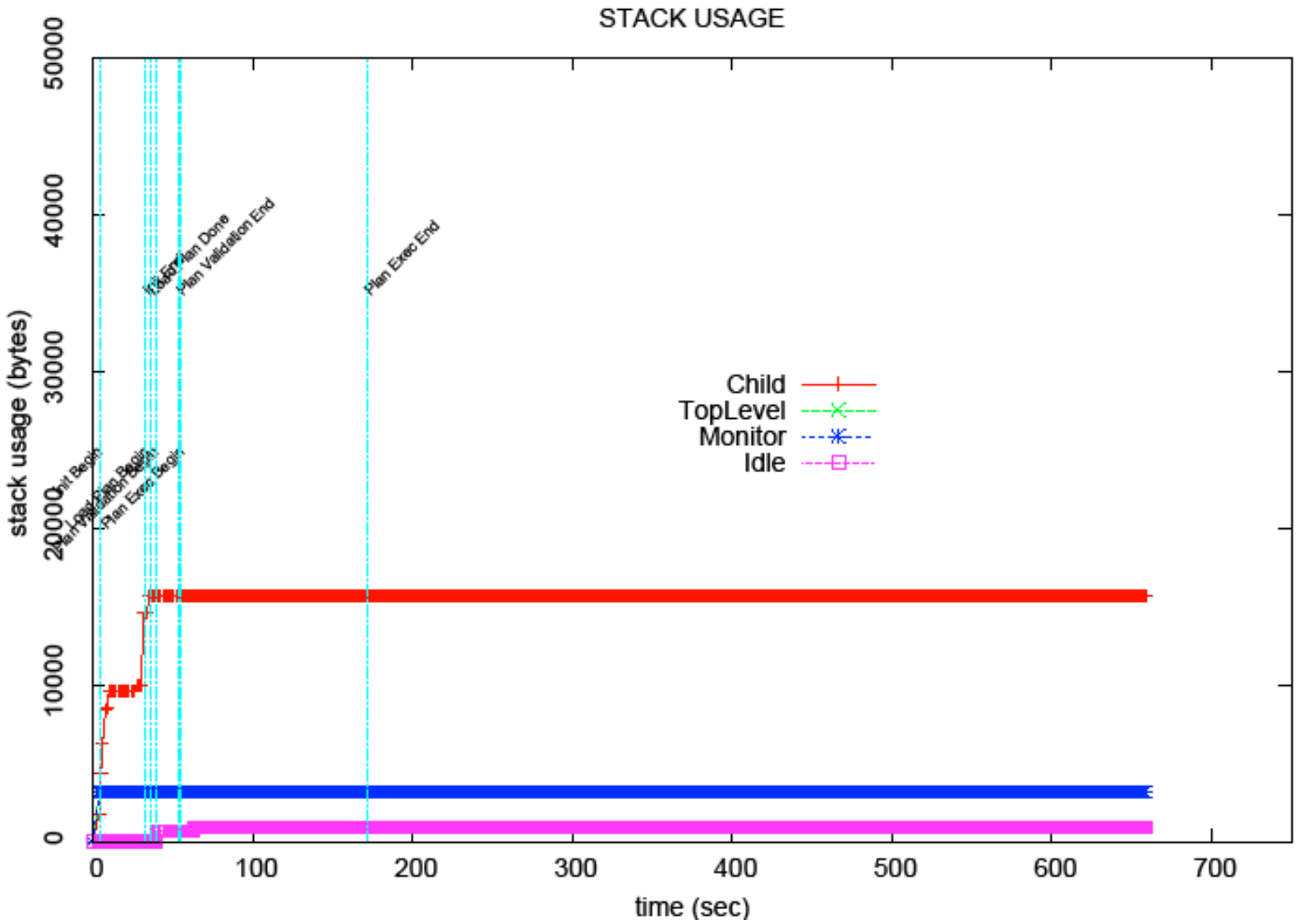
Memory usage: orbiter (LEON3)



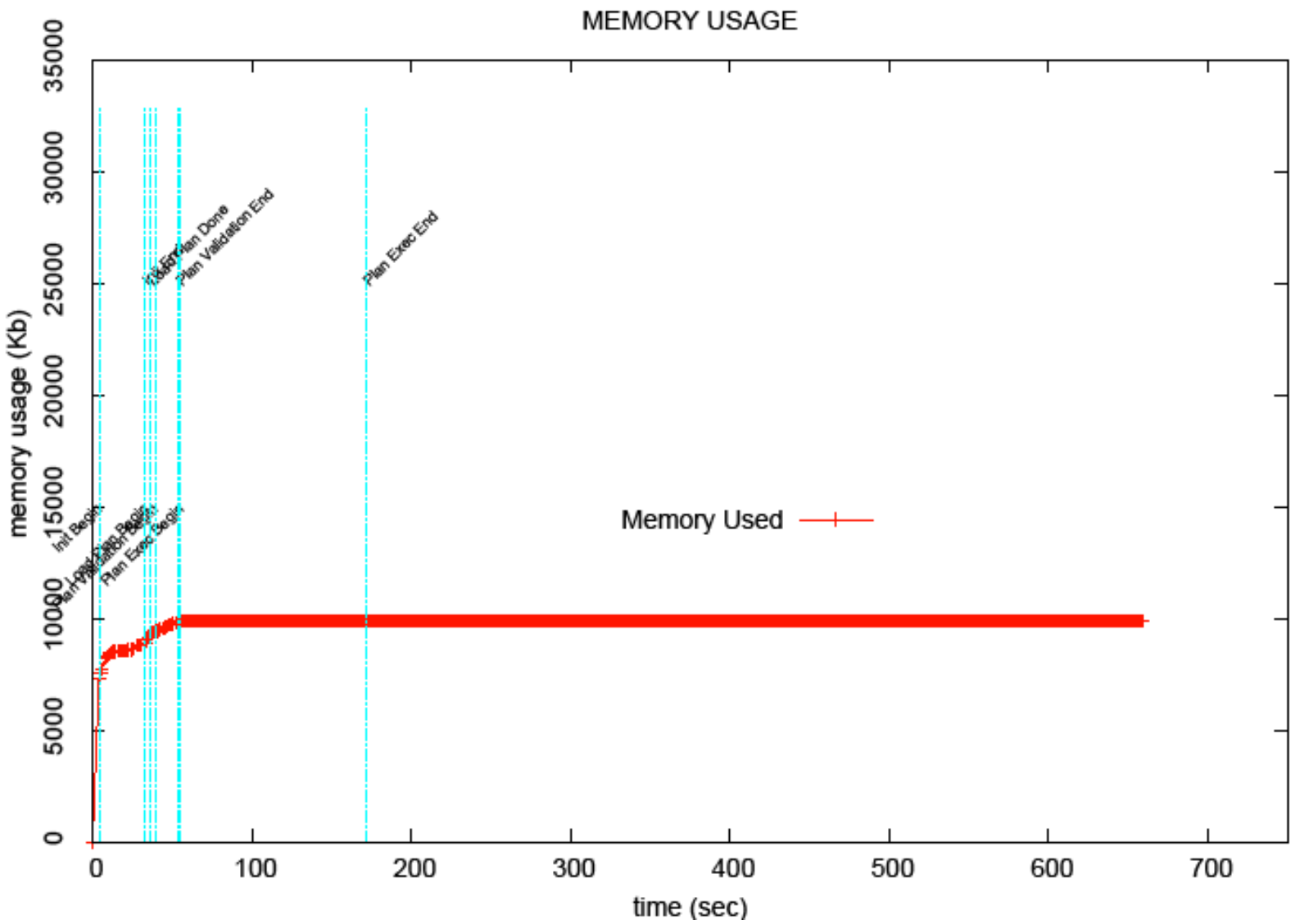
CPU Usage: small rover (ERC32)



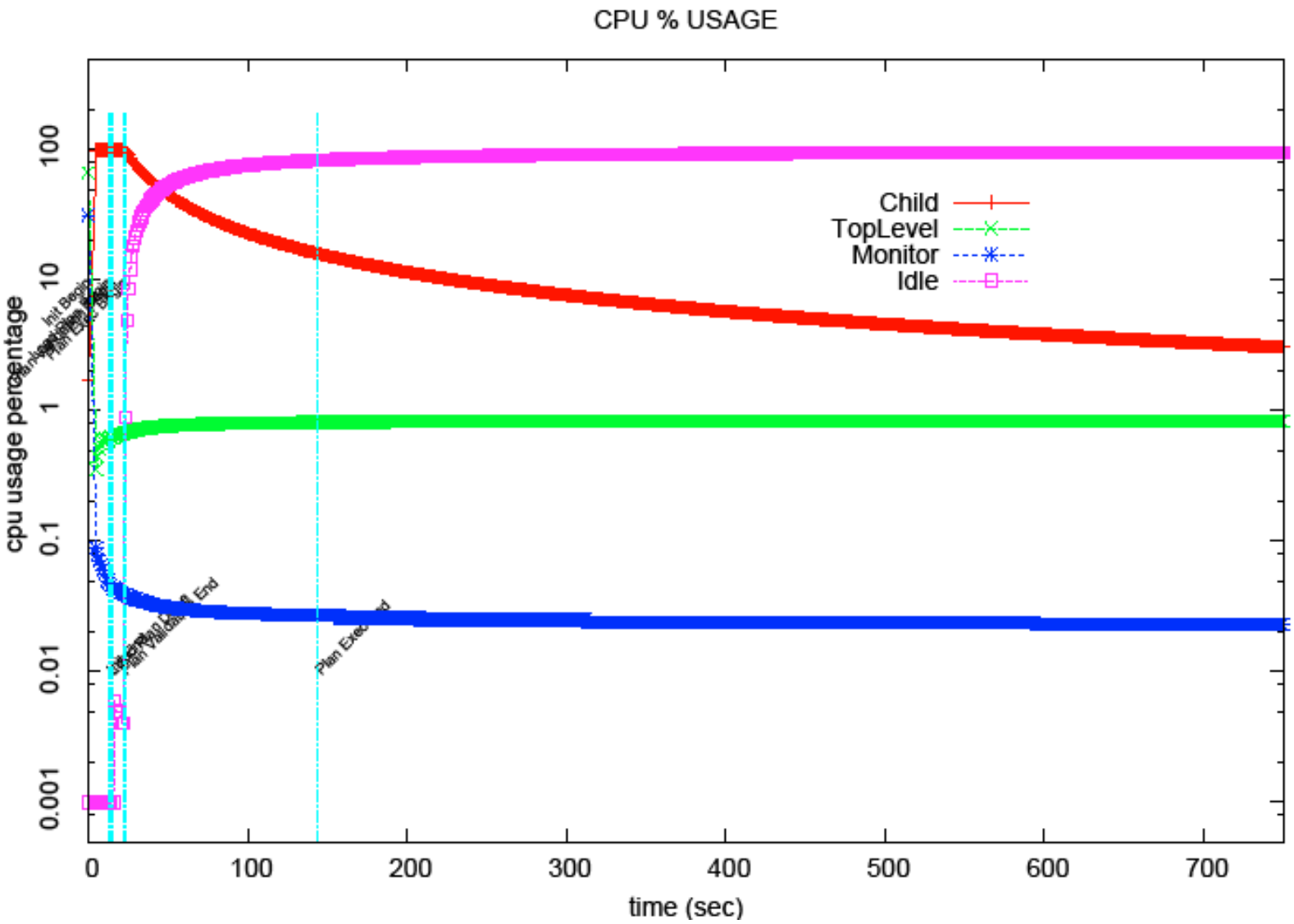
Stack usage: small rover (ERC32)



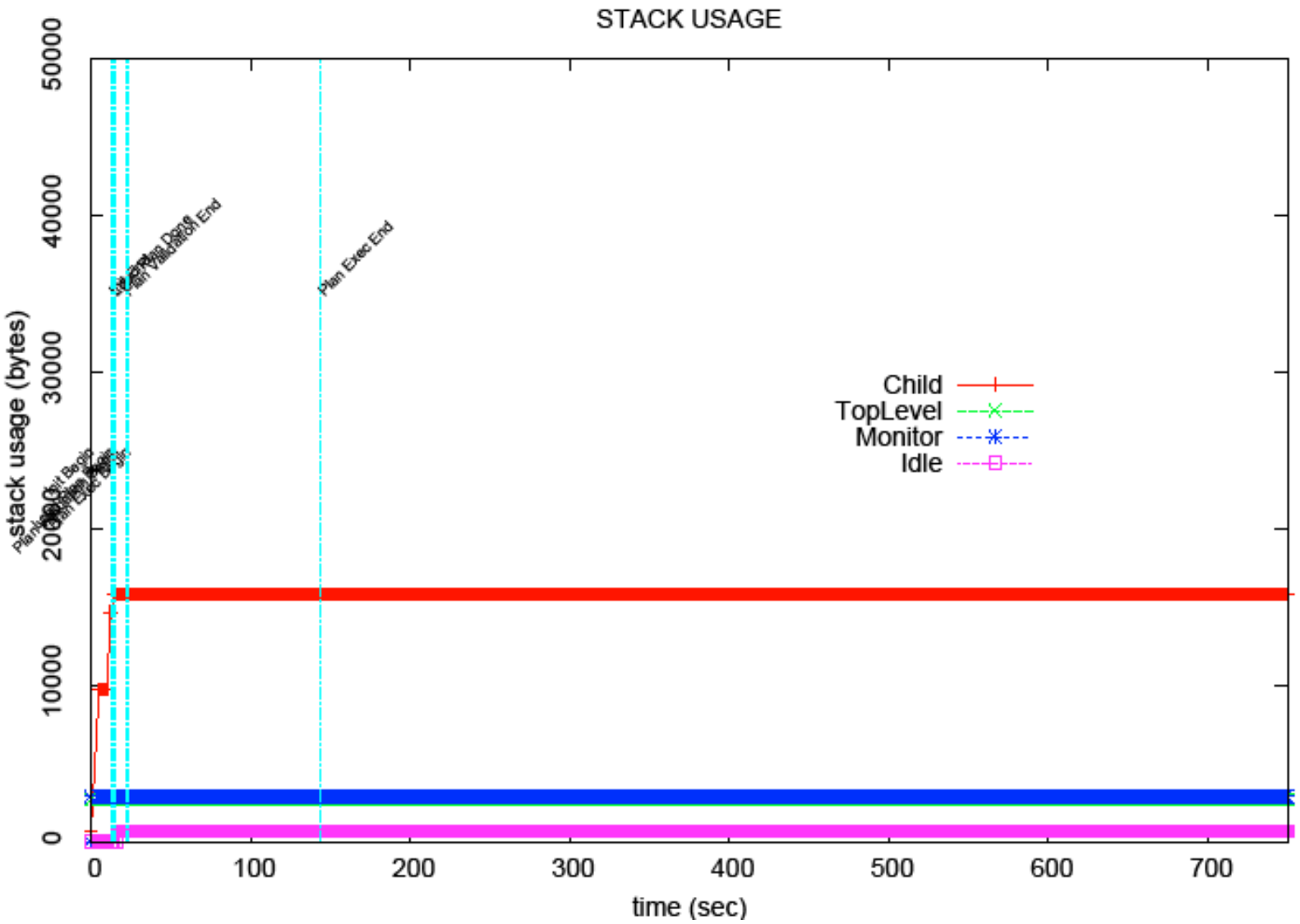
Memory usage: small rover (ERC32)



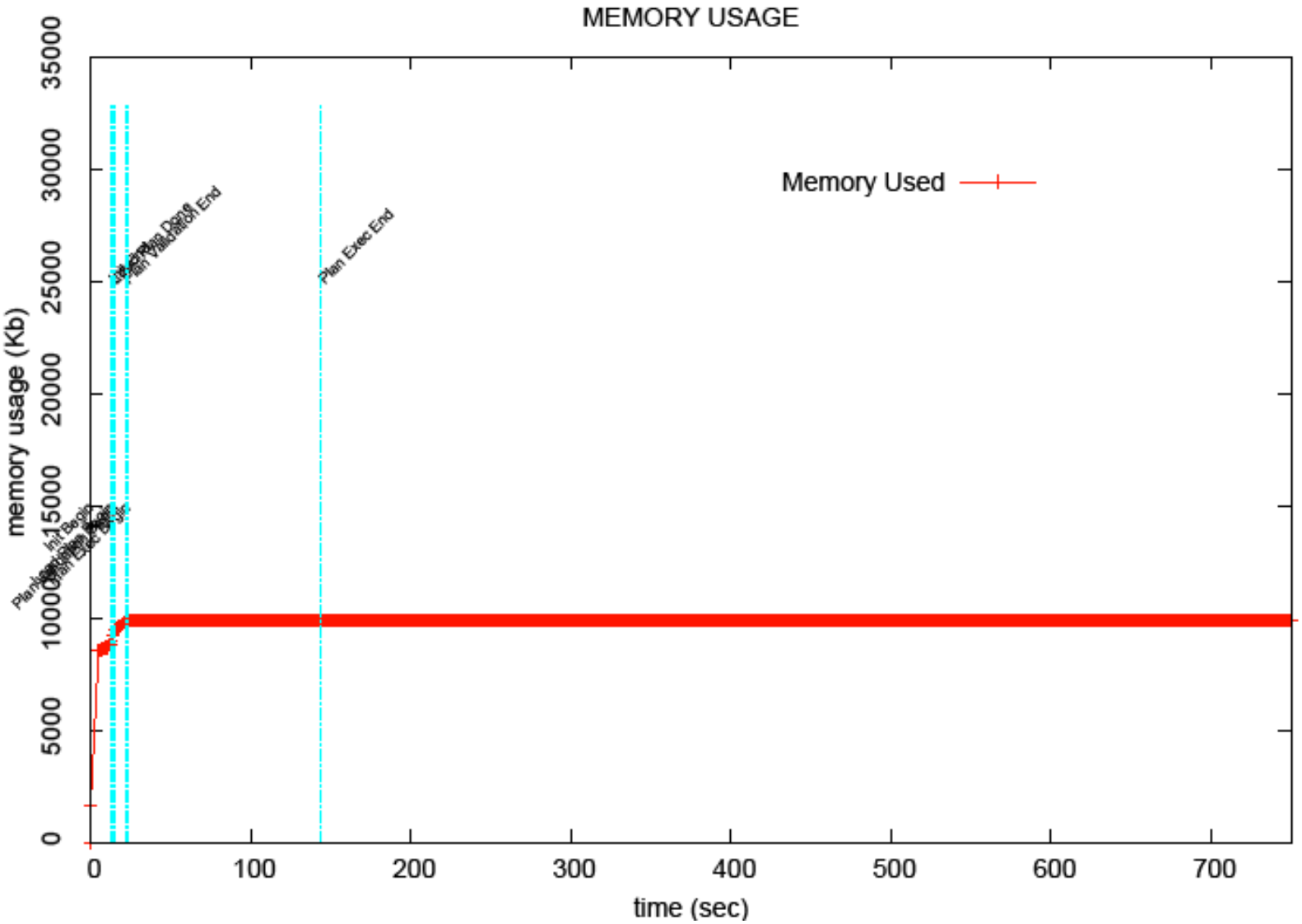
CPU Usage: small rover (LEON3)



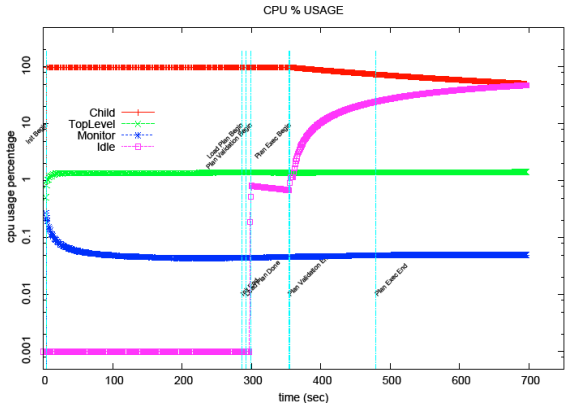
Stack usage: small rover (LEON3)

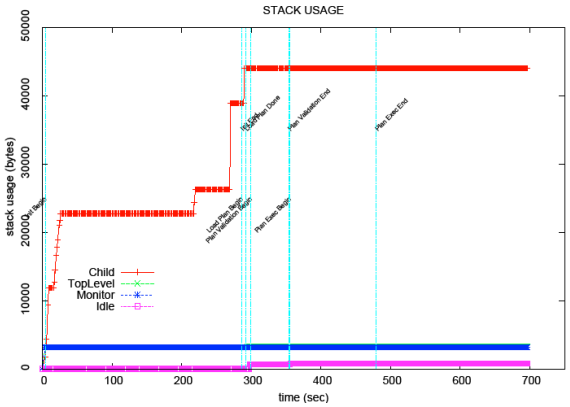


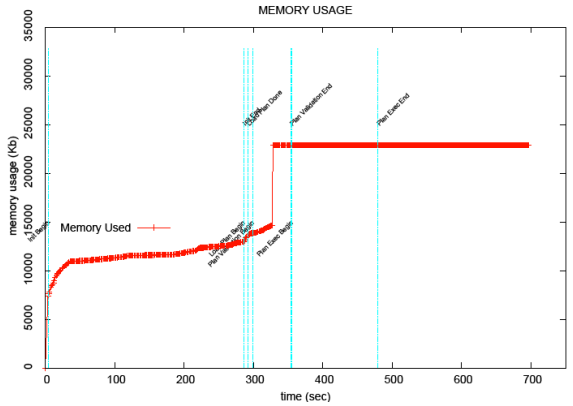
Memory usage: small rover (LEON3)

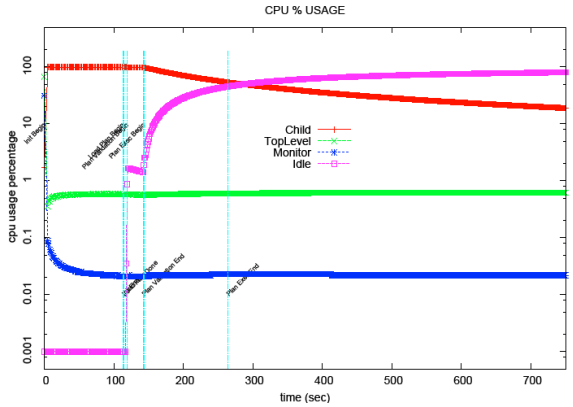


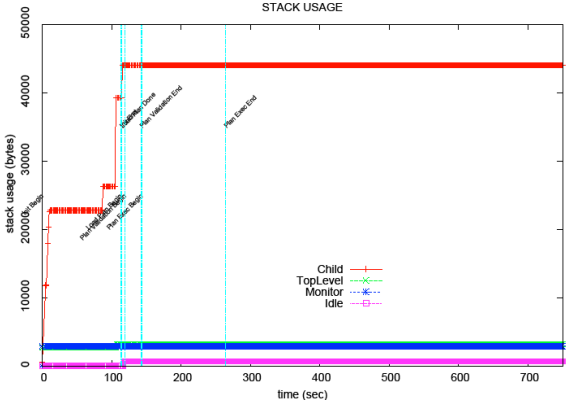
ERC32 vs LEON3 (rover)

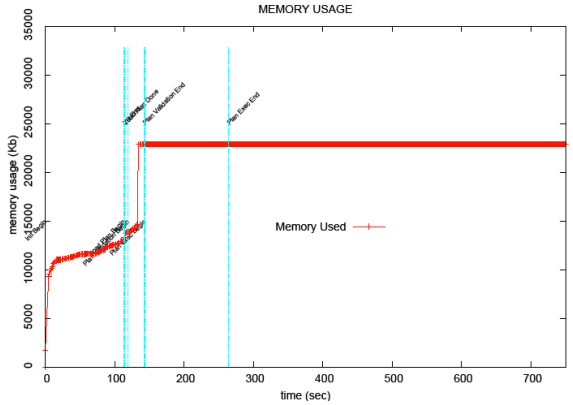




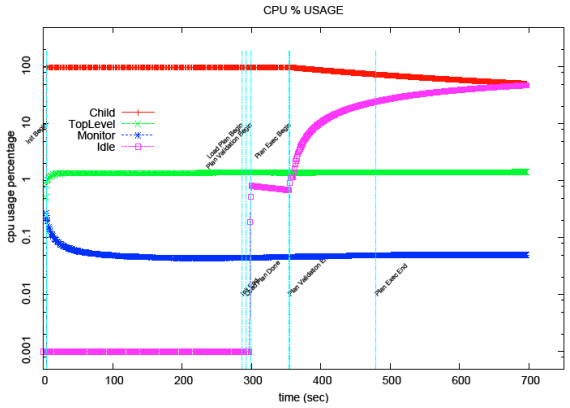


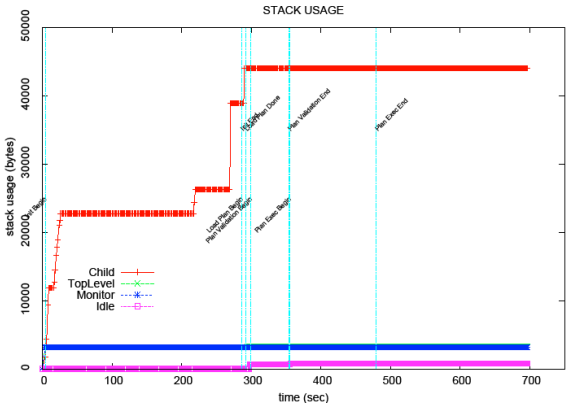


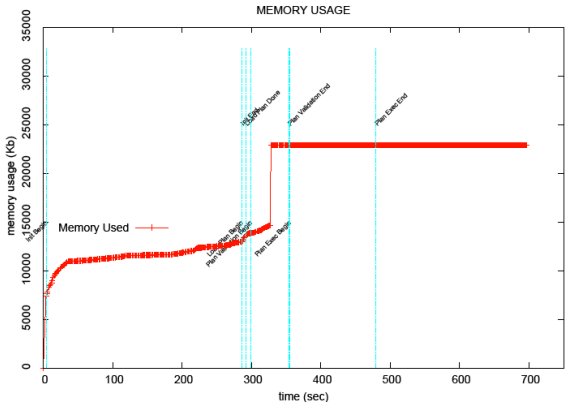


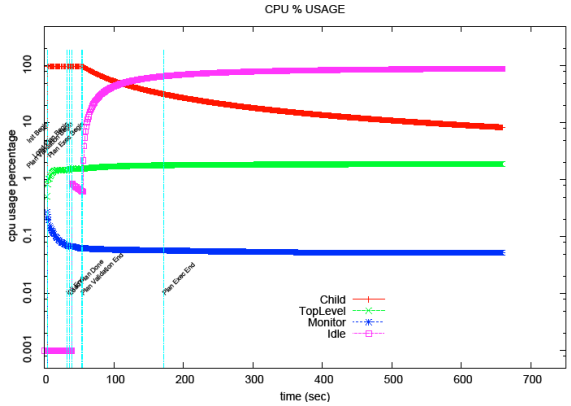


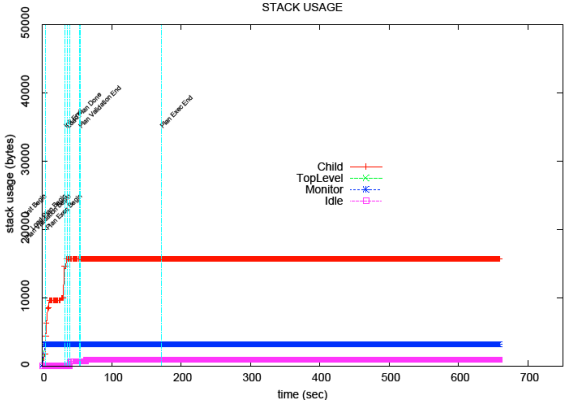
Rover vs Small Rover (ERC32)

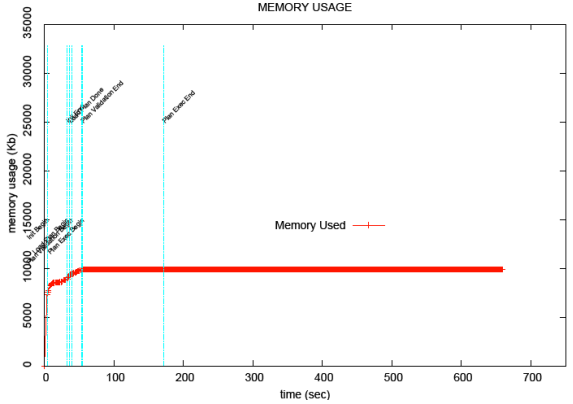












Time x Activity Summary

ERC32	Initialization	Load Plan	Plan Validation	Plan Execution	Plan Generation
Rover	282s	6s	55s	125s	1349s
Small Rover	33s	3s	15s	116s	87s
Orbiter	9s	2s	1s	16s	6s

LEON3	Initialization	Load Plan	Plan Validation	Plan Execution	Plan Generation
Rover	113s	2s	23s	121s	540s
Small Rover	13s	1s	6.5s	121s	34s
Orbiter	1s	0.5s	1s	16s	2s

Comments

- Reported Plan Execution time is elapsed
  - computation time in ARE below 5%
- If more CPU intensive tasks running at the same time on the CPU
  - Running time can increase
  - CPU percentage usage can reduce
- Rover required slightly less than 32Mb for plan generation.

Industrial Perspective

Andrea Guiotto

Industrial perspective

- Maturity of ARE
- Use in real applications
- Impact on the currently used architectures
- Impact on the development process
  - Formal techniques to support the development
- Impact on future space architectures
  - Potential reduction of costs of some HW
- Integration with the spacecraft simulator

Maturity of ARE

- The approach integrates in a unique framework different aspects such as planning, monitoring, execution, re-planning
- The formal model captures the possible non-determinism in commands effects and environment

Maturity of ARE (II)

- Formal model contains the status of the controlled plant and of environment that is partially observable due to the limited availability of sensors.

- ARE finds a plan that will guarantee the achievement of a goal despite the non determinism in the initial condition and partial observability of controlled plant.

Maturity of ARE (III)

- ARE approach is suitable w.r.t.
  - Mission characteristic (unpredictable local conditions)
  - Mission constraints (limited bandwidth, intermittent visibility, long round trip delay)
  - System operations (perform measurement cycle)
- ARE is suitable to be embedded in current on-board computers.
- The dimension of code could be optimized.
- Performance and re-planning ability depend on the complexity of model
- Performances of ARE remains better under workstation Linux/x86 than in RTEMS/Leon3

Use in real applications

- ARE has been developed as a potential BB to be reused in on-board software
  - ARE is robust to stack overflow
  - ARE does not use unbounded depth recursive functions.
  - No dynamic memory allocation.
  - It can receive commands from ground.
- ARE guarantees the safety of system:
  - In case it does not find the plan it remains in a safe state waiting for ground intervention.
- ARE can be used as ground support tool for mission analysis.

Impact on real applications

- In order to integrate ARE it is necessary to respect its interfaces:
  - ARE SW has been developed on top of the POSIX 1003.1B API
  - Low level routines must interface sensors and actuators
- The use of ARE could require to improve the performance of microprocessors.

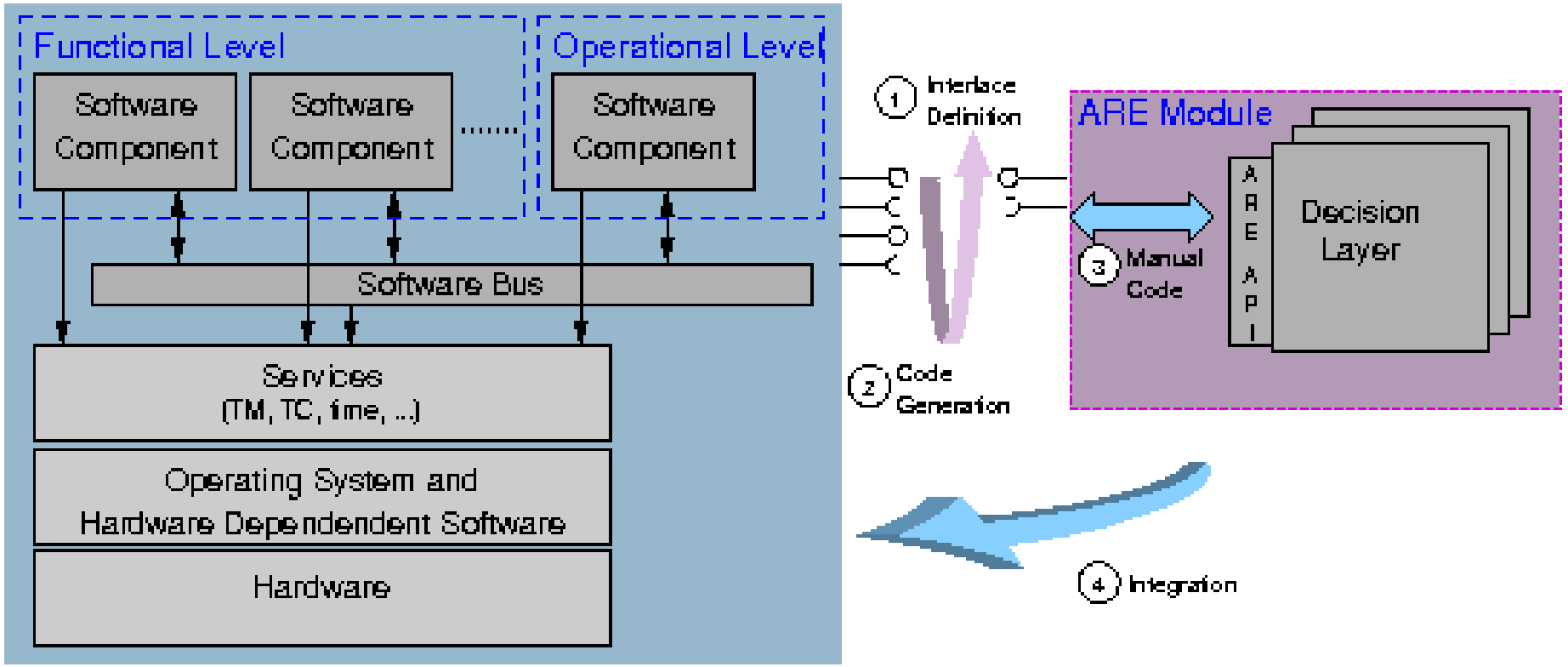
Impact on development process

- Development of system model in SMV from Matlab/Stateflow/Simulink or textual description
- Integration phase
- Model Checker NuSMV should be certified as space software

Future space architectures

- Investigate software replacement of some HW component dedicated to autonomy
  - This can increase the criticality level of software
  - Cost of software validation campaign could increase

Integration with the simulator



Integration with the simulator (II)

- Spacecraft simulator is based on a library that contains the basic common functionalities (TC management, TM management, software bus services).
- The services offered by this library allow to easily integrate software components into a software bus.
- The mock-up is running on a Linux PC host with a simulator (SiS64 on ERC32)
- Commanding (TC, scripts) as well as display and reporting (TM, reports, checks) are network transparent and support multiple users on the same simulation.

Integration with the simulator (III)

- ARE has been converted in a simulator component as Ostrales task:
  - POSIX API of simulator are not equal to RTEMS POSIX API
- Increase RAM and ROM size in the linker script.

Discussion, Related Work, and Future Perspectives

Alessandro Cimatti

Scientific Perspective

- A comprehensive approach
  - all functions in a unique framework
    - » plan generation, execution monitoring, FDIR, replanning
  - a unique model for all functions
  - a symbolic approach
- An expressive framework
  - nondeterministic partially observable domains
  - working with assumptions
  - linking reasoning about resources and logical reasoning
    - » logic vs computation
- Platform characterization
  - results are not dramatic (promising?)

By product: use for ground ops

- Adaptation for ground operation
  - Used to generate/validate plans to be uploaded
  - could be a hand-written or mixed-initiative plan
  - depends on ground information
    - » available? accurate?
- Can be used for explanatory/diagnostic purposes
  - to reconstruct on-board behaviour
  - diagnosis on ground
- Uniform framework can be shared between on-board and ground

Comparison with existing approaches

Comparison with existing approaches

- The Remote Agent NASA experience
  - goal-directed planning, scheduling and diagnosis
  - model-based executive Livingstone
  - key differences
    - » several formalisms
    - » explicit state search
- MUROCO-II
  - generic framework for the specification and verification of space missions
  - emphasis on off line vs on board
    - » potentially complementary
  - emphasis on model/plan validation rather than plan generation
    - » unable to deal with partial observability, non determinism, and diagnosis
    - » unable to generate and monitor execution of plans
- MMOPS
  - Timeline Verification, Control and Repair
  - on-board approach
  - limited expressiveness, emphasis on scheduling aspects
  - unable to deal with generic models

Comparison with existing approaches

- The Remote Agent NASA experience
  - goal-directed planning, scheduling and diagnosis
  - model-based executive Livingstone
  - key differences
    - » several formalisms
    - » explicit state search
- MUROCO-II
  - generic framework for the specification and verification of space missions
  - emphasis on off line vs on board
    - » potentially complementary
  - emphasis on model/plan validation rather than plan generation
    - » unable to deal with partial observability, non determinism, and diagnosis
    - » unable to generate and monitor execution of plans
- MMOPS
  - Timeline Verification, Control and Repair
  - on-board approach
  - limited expressiveness, emphasis on scheduling aspects
  - unable to deal with generic models

Comparison with existing approaches

- The Remote Agent NASA experience
  - goal-directed planning, scheduling and diagnosis
  - model-based executive Livingstone
  - key differences
    - » several formalisms
    - » explicit state search
- MUROCO-II
  - generic framework for the specification and verification of space missions
  - emphasis on off line vs on board
    - » potentially complementary
  - emphasis on model/plan validation rather than plan generation
    - » unable to deal with partial observability, non determinism, and diagnosis
    - » unable to generate and monitor execution of plans
- MMOPS
  - Timeline Verification, Control and Repair
  - on-board approach
  - limited expressiveness, emphasis on scheduling aspects
  - unable to deal with generic models

Dissemination

- Project Web site
  - http://es.fbk.eu/projects/esa_omc-are
- Accepted presentations
  - **On-Board Autonomy via Symbolic Model Based Reasoning**, by *M. Bozzano*, *A. Cimatti*, *A. Guiotto*, *A. Martelli*, *M. Roveri*, *A. Tchaltsev* and *Y. Yushtein*. In 10th ESA Workshop on Advanced Space Technologies for Robotics and Automation (ASTRA'2008). 11 - 13 November 2008, ESA/ESTEC, Noordwijk, The Netherlands.
  - **On Board Model Checking for Space Applications**, by *A. Cimatti*, *A. Guiotto* and *M. Roveri*. In ESA Workshop on Avionics Data, Control and Software Systems (ADCSS). 29 - 31 October 2008, ESA/ESTEC, Noordwijk, The Netherlands.
- Intended submissions
  - Major aero-space conferences: DASIA, iSAIRAS
    - » case studies and characterization
  - Major AI conferences: IJCAI, ICAPS, ECAI
    - » extensions to planning techniques
  - Formal methods related conferences: ISOLA
    - » formal methods at work on "non-standard" applications
  - Major AI journals: Artificial Intelligence, Journal of Art. Int. and Research (JAIR)
    - » comprehensive project description

Conclusions

Results of the study:

- A unique and comprehensive framework encompassing different autonomy aspects
  - plan generation, validation, monitoring, execution and FDIR
- Implementation using symbolic techniques
- Case studies
- Experiemental characterization
  - functional and on platform

Open and Future Directions

- Further technology push
  - Footprint reduction
    - » customize nusmv library for embedded
  - Memory / CPU usage
    - » Cone-of-influence reduction
    - » Alternate heuristics
    - » BDD partitioning
    - » SAT-based techniques
    - » Adaptation/use of classical planning techniques
- Model-based Validation of Intelligence
  - proof of correctness of conceptual framework
  - validation of implementation
    - » "translation validation" approach
    - » independent checking of generated plan
- Improve accuracy in reasoning
  - Resource model
    - » beyond bounded arithmetics?
  - Diagnosis
    - » need for finer-grain probabilistic reasoning?

Future work

- Define process and extend tool support
  - Build automatic translator of NuSMV models from Matlab/Simulink/Stateflow models.
  - Low level routines can be derived directly from the code automatically generated with Matlab Real Time Workshop
- Strong connection with the COMPASS (Correctness, Modeling and Performance of Aerospace Systems) project sponsored by ESA
  - http://compass.informatik.rwth-aachen.de/
  - Same technology used in two projects
- Explore continuous ground/on board spectrum

DEMO

Andrea Guiotto, Marco Roveri

OPEN DISCUSSION