Modelling Software-based Systems

Lecture 6

Validation, Verification and Proof Tools for Event-B for Correct-by-Construction

Telecom Nancy (IL et LE)

Dominique Méry
Telecom Nancy,Université de Lorraine

8 septembre 2026

dominique.mery@loria.fr

General Summary

1. Verification
2. Proof Management
3. Development Methodology and Tools for Correct By Construction
4. Project Management

Current Summary

1. Verification
2. Proof Management
3. Development Methodology and Tools for Correct By Construction
4. Project Management

Summary

1. Verification
2. Proof Management
3. Development Methodology and Tools for Correct By Construction
4. Project Management

Annotation of programs

$$\begin{array}{l} \ell : \{P_\ell(v)\} \\ \quad cond_{\ell,\ell'}(v) \longrightarrow v := f_{\ell,\ell'}(v) \\ \ell' : \{P_{\ell'}(v)\} \end{array}$$

$$\begin{array}{l} \ell_0^1 : \{x = 0\} \\ \quad x := x + 1; \\ \ell_0^1 : \{x = 1\} \end{array}$$

$$\begin{array}{l} e(\ell, \ell') \\ \quad \textbf{WHEN} \\ \quad\quad c = \ell \\ \quad\quad cond_{\ell,\ell'}(v) \\ \quad \textbf{THEN} \\ \quad\quad c := \ell' \\ \quad\quad v := f_{\ell,\ell'}(v) \\ \quad \textbf{END} \end{array}$$

- v is the state meory variable or list of memory variables ; v includes the local variables and the results variables.
- c is a new variable which is modelling the control flow and its type is LOCATIONS.
- $e(\ell, \ell')$ is simulating the computation flow starting from ℓ and moving to ℓ' ; v is updated.

From annotations to invariants

INVARIANTS

$inv_i : c \in \text{LOCATIONS}$
$inv_j : v \in Type$
$\ldots$
$inv_k : c = \ell \Rightarrow P_\ell(v)$
$inv_m : c = \ell' \Rightarrow P_{\ell'}(v)$
$\ldots$
$th_n : A(c, v)$

- $Type$ is the type of the variables v and is a set of possible values defined in the context C.
- The annotation is giving us for free the conditions satisfied by v when the control is in ℓ, (resp. in ℓ').
- $A(c, v)$ is a safety property that we are supposed to check and the case of Event-B, it is a theorem.

Partial correctness using Event-B models

For each pair of successive labels ℓ, ℓ', the three statements are equivalent :

- $P_\ell(v) \wedge cond_{\ell,\ell'}(v) \wedge v' = f_{\ell,\ell'}(v) \Rightarrow P_{\ell'}(v')$
- $I(c, v) \wedge c = \ell \wedge cond_{\ell,\ell'}(v) \wedge c' = \ell' \wedge v' = f_{\ell,\ell'}(v) \Rightarrow (c' = \ell' \Rightarrow P_{\ell'}(v'))$
- $I(c, v) \wedge BA(e(\ell, \ell'))(c, v, c', v') \Rightarrow (c' = \ell' \Rightarrow P_{\ell'}(v'))$

L

et AA an annotated algorithm with precondition **pre**(AA)(v) and postcondition **post**(AA)(v_0,v). Let the context C and the machine M generated from AA using the construction given previously. We assume that ℓ_0 is the first label and ℓ_e is the last label. We add the following safety properties in the machine M :

- $c = \ell_0 \wedge$ **pre**$(\mathrm{AA})(v) \Rightarrow P_{\ell_0}(v)$
- $c = \ell_e \Rightarrow (P_{\ell_e}(v) \Rightarrow$ **post**$(\mathrm{AA})(v_0, v)$

If proof obligations are discharged, then the annotated algorithm AA is partially correct with respect to ist pre/post specification.

Summary

1. Verification
2. Proof Management
3. Development Methodology and Tools for Correct By Construction
4. Project Management

Current Summary

1 Verification

2 Proof Management

3 Development Methodology and Tools for Correct By Construction

4 Project Management

Proof Management

- The panel called *proof control* is providing tools for *discharging* proof obligations
- Proof obligations are stated as follows : $\Gamma \vdash P$: *When assertions of* Γ *are supposed to hold, then* P *does hold*
- Problem : Checking that the sequent $\Gamma \vdash P$ is valid.
- How : Applying automatic procedures as pp, pr etc
- ... or helping the proof tool to build the *proof tree*

Main Limits of PP (Predicate Prover)

- Quite sensitive to useless hypotheses.
- Unable to automatically abstract some statements.
- Sometimes poor management of equalities.
- Needs some creative hints

Creative Hints

- Proposing clever lemma : the user may have interesting ideas because he/she knows the problem to solve.
- Suggesting a proof by case.
- Proposing a witness for an existential goal.
- Proposing a universal hypothesis instanciation.
- etc.

Sequents and Inference rules

- Sequent (where $list_of_hypotheses$ might be empty).

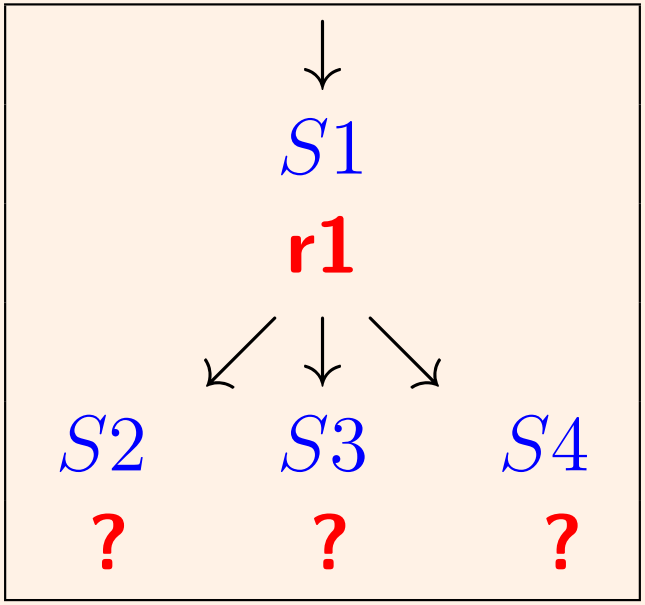
$$\boxed{list_of_hypotheses \quad \vdash \quad conclusion}$$

- Inference rule (where $list_of_sequents$ might be empty).

$$\boxed{\frac{list_of_sequents}{sequent}}$$

- Backwards application of inf. rules $\rightsquigarrow$ proof tree.

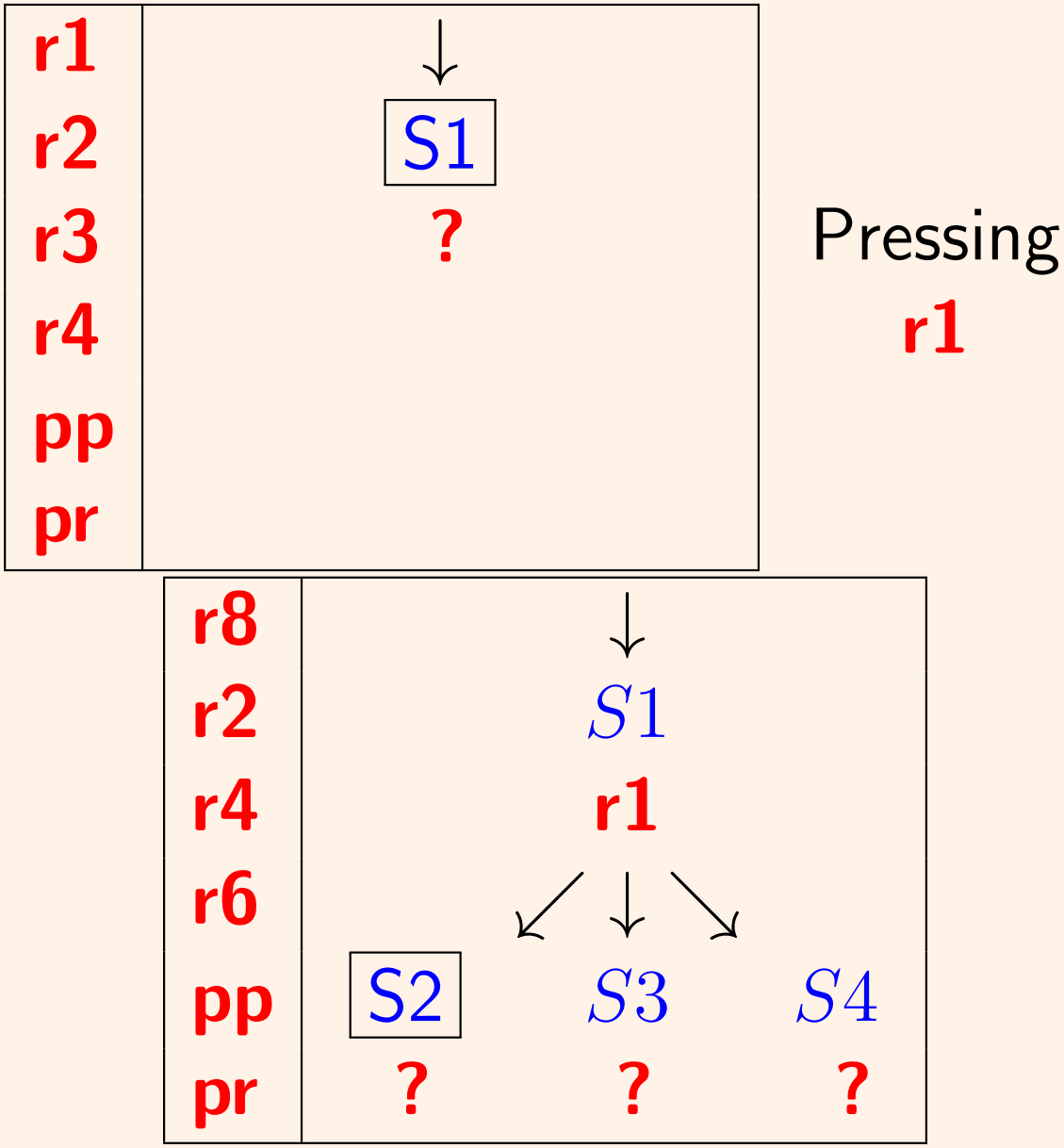
The Proof Tree



- Rule **r1** is applied to sequent S1.

- Producing sequents S2, S3, and S4.

Interface : Tree and Palette of Rule Buttons



- The proof tree is constructed interactively.

Difficulty : Size of Window

r8			
r2	$\boxed{S2}$	$S3$	$S4$
r4	?	?	?
r6			
pp			
pr			

pp
then
r6

r5			
r4	$S2$	$S3$	$S4$
r6	pp	r6	?
r9			
pp	$\boxed{S5}$	$S6$	
pr	?	?	

- Our tentative "interface" does not work.

More Precise Form of Sequent

$$H \ \vdash \ L \implies G$$

- Most of the time the conclusion has an implicative form
- G is called the goal (it is a non-conjunctive predicate)
- L are called the local hypotheses (might be empty)

$$H \ \vdash \ I(x) \ \wedge \ G(x) \ \wedge \ P(x, x') \ \implies \ I(x')$$

Tentative Solution : Using Two Windows

- Tree window with simplified sequents (goals only)

- Sequent window containing the

full sequent of interest

$$H2 \vdash L2 \Longrightarrow G2$$

r2 r4 r6 pp pr

Sequent Window

$$G1$$

r1

$$\swarrow \downarrow \searrow$$

$$\boxed{G2} \quad G3 \quad G4$$

? ? ?

Tree Window

One Proof Step

$H2 \vdash L2 \Longrightarrow G2$
r2 r4 r6 pp pr

$G1$

r1

$\swarrow \downarrow \searrow$

$\boxed{G2}$	$G3$	$G4$
?	**?**	**?**

Pressing **pp**

$H3 \vdash L3 \Longrightarrow G3$
r2 r5 r8 pp pr

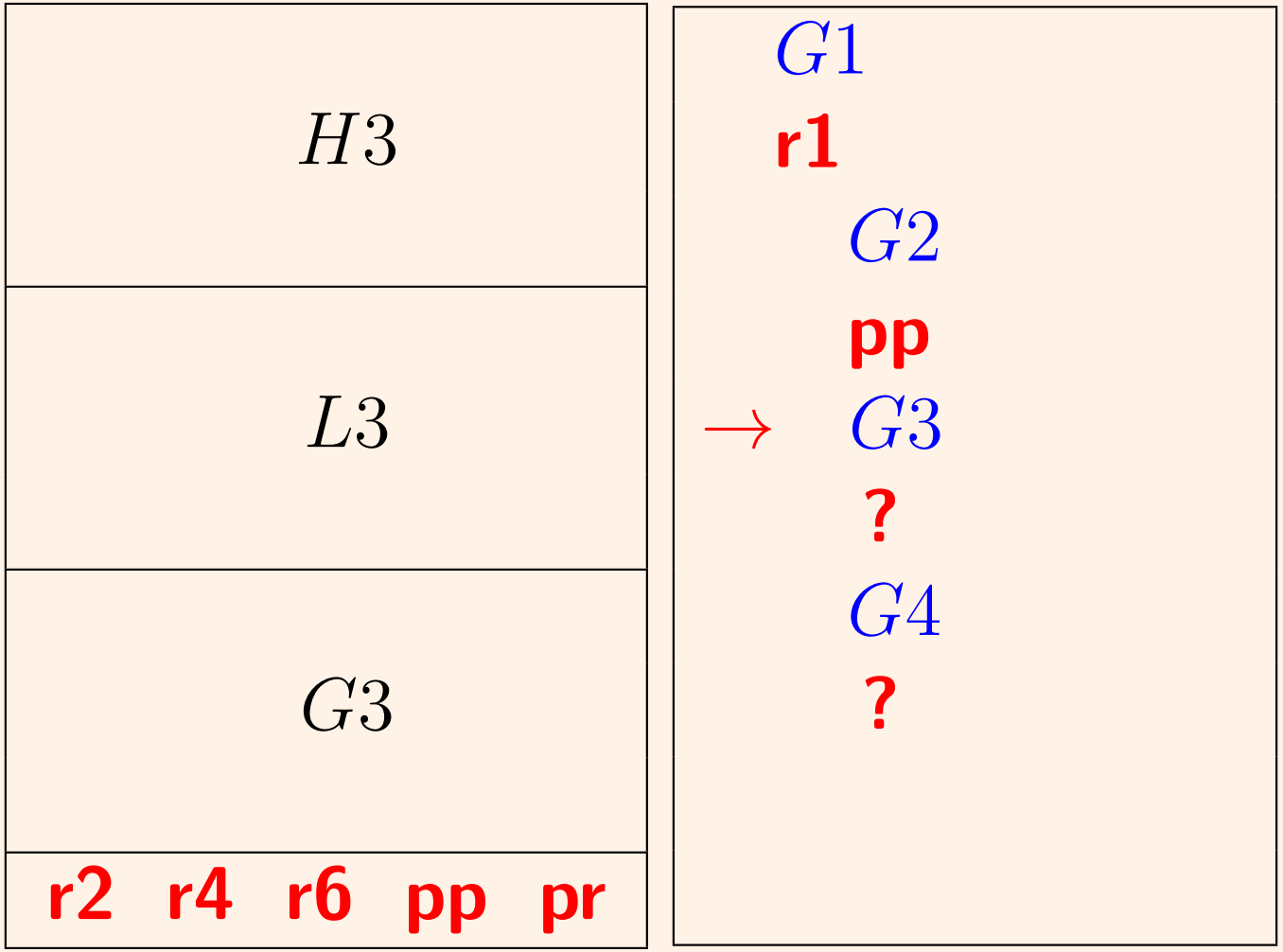
$G1$

r1

$\swarrow \downarrow \searrow$

$G2$	$\boxed{G3}$	$G4$
pp	**?**	**?**

More Realistic Windows



More Elaborate Sequents

$$\underbrace{hidden \;\; ; \;\; searched \;\; ; \;\; cached}_{list_of_hypotheses} \;\; \vdash \;\; \underbrace{local \;\; \Longrightarrow \;\; goal}_{conclusion}$$

$hidden$	Not visible on the sequent window
$searched$	Visible after some search in $hidden$
$cached$	Visible but not part of the conclusion any m
$local$	Visible and part of the conclusion

Current Summary

1 Verification

2 Proof Management

3 Development Methodology and Tools for Correct By Construction

4 Project Management

Context and Objectives

Context

Developing life-cycle methodology and a set of associated techniques and tools to develop the highly critical systems using formal techniques from requirements analysis to code generation.

Objectives

- To establish a unified theory for the critical system development.
- To Build a set of tools for supporting new development life-cycle methodology.
- To develop a closed-loop system for verification purpose.
- Graphical based refinement technique to handle the complexity of the system.
- To satisfy requirements and metrics for certifiable assurance and safety.
- To support evidence-based certification.

Main Contributions

Development Methodology

- Formal Methods based Critical System Development Life-Cycle

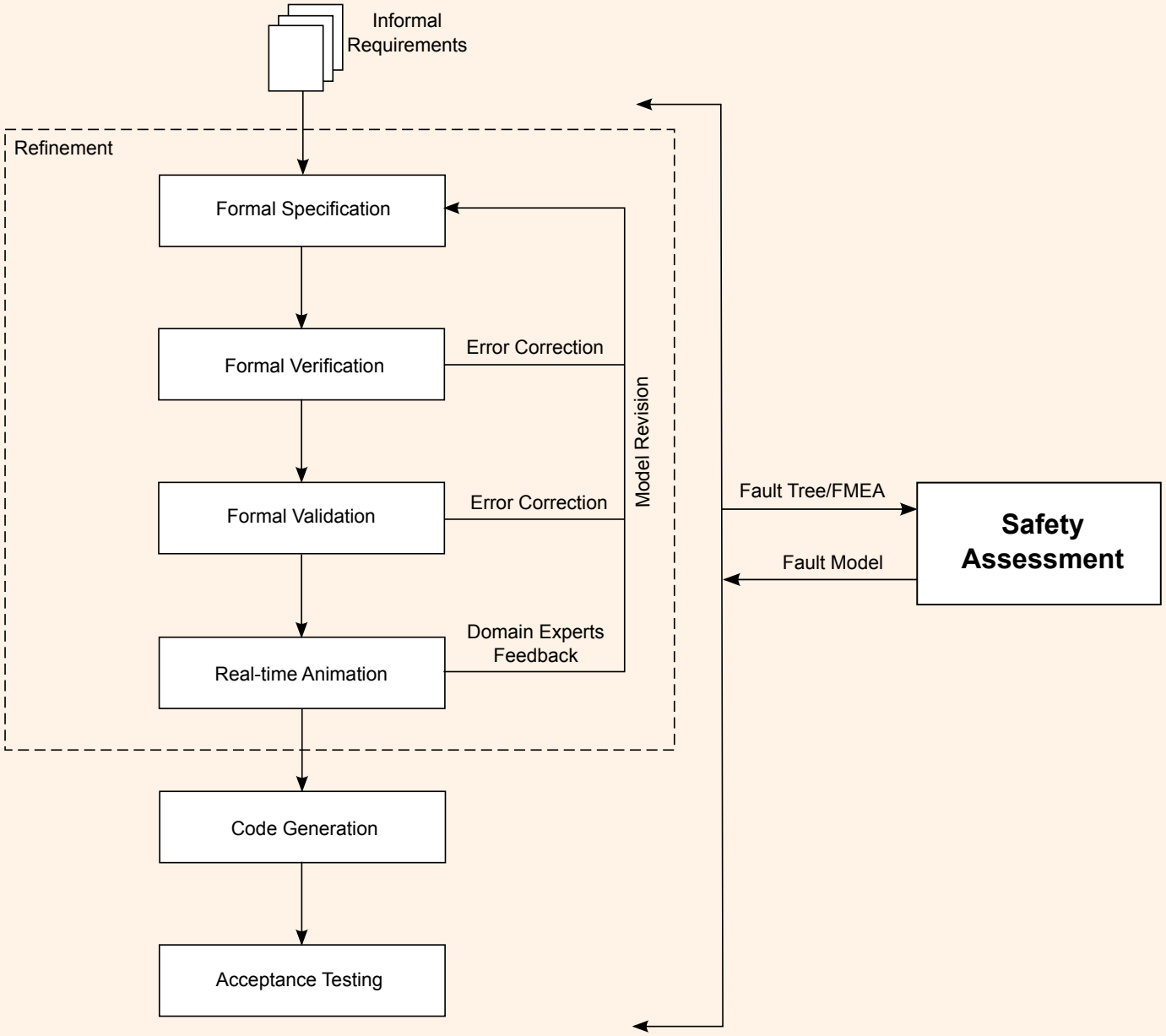
Techniques and Tools

- Real-Time Animator
- Refinement Chart
- Automatic Code Generator : EB2ALL
- Formal Logic Based Heart-Model

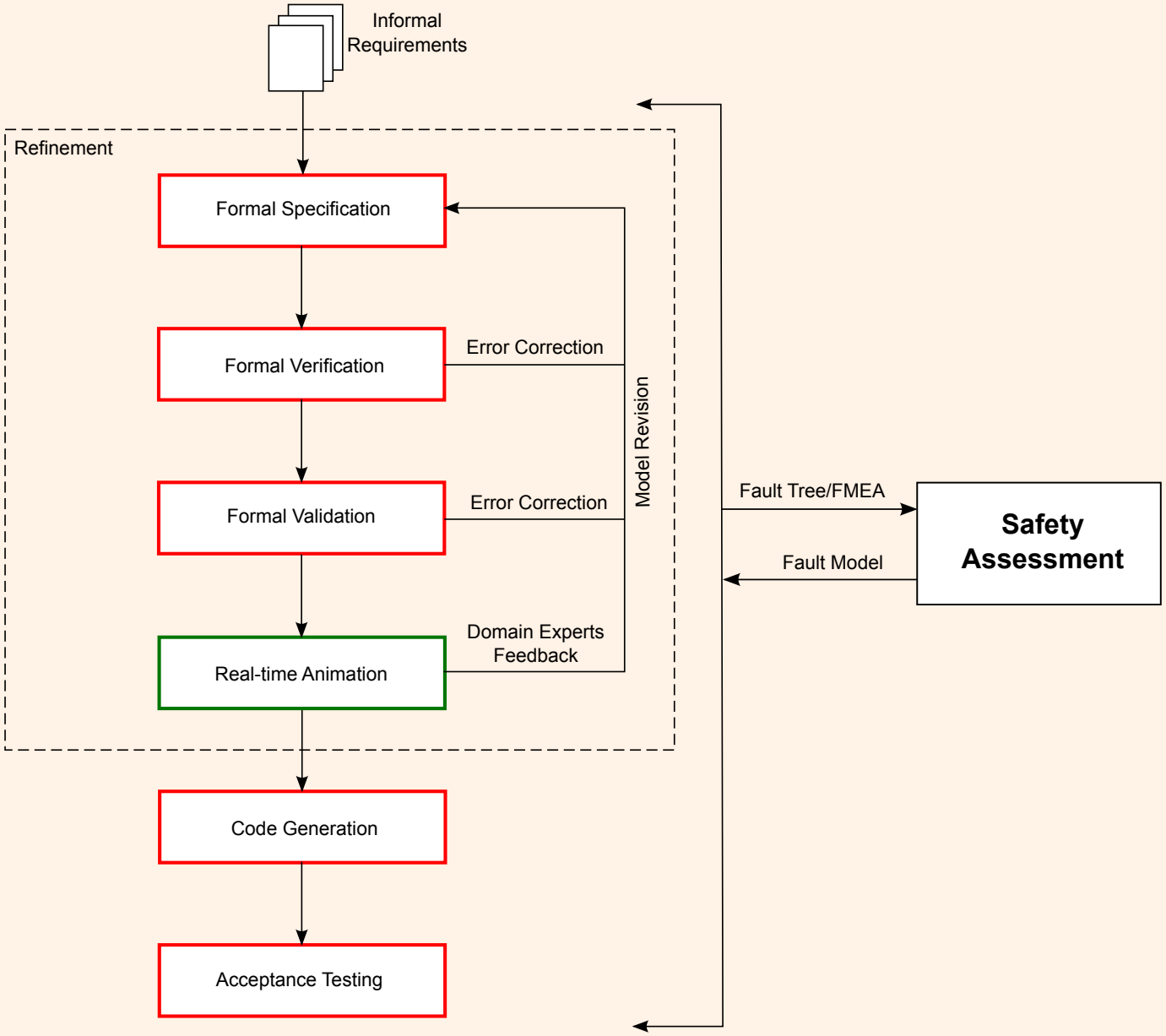
Case Studies

- The Cardiac Pacemaker

Critical System Development Life-Cycle Methodology



Critical System Development Life-Cycle Methodology



Real-Time Animator

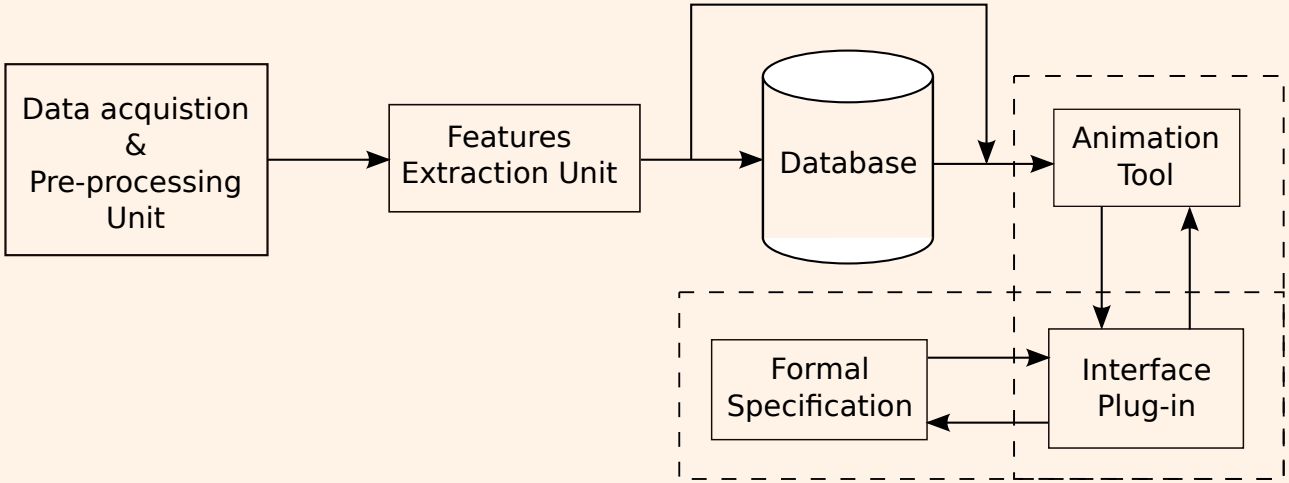
What is Real-time Animator ?

- Visual representation of formal model using real time data set.

Why should we use Formal Model Animator ?

- To validate system behavior according to the stakeholders
- To express formal models for non-mathematical domain experts
- To discover the error in the early stage of system development (Traceability)

Proposed Architecture



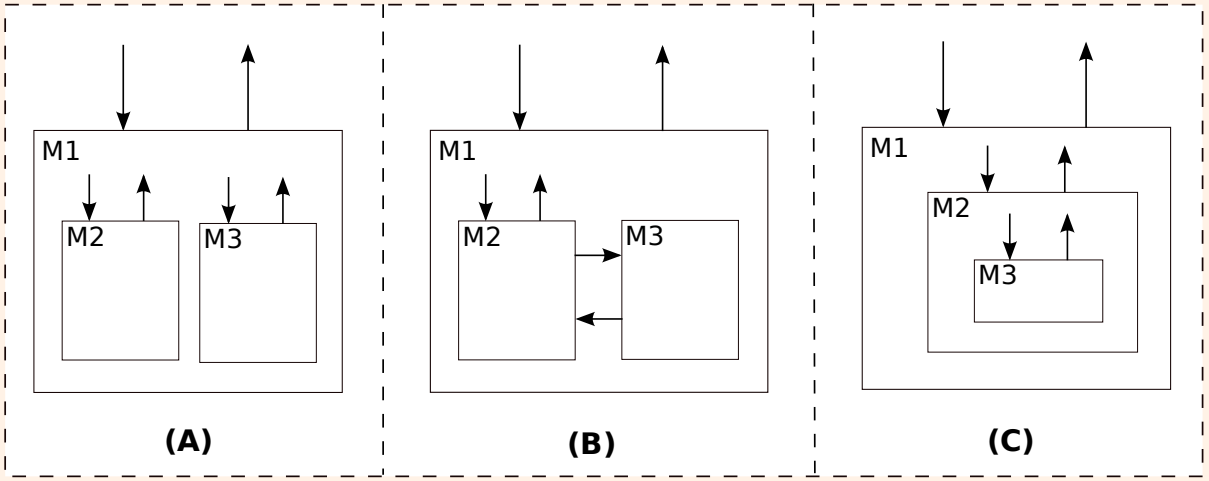
Refinement Chart

- Refinement chart is a graphical representation of a complex system using layering approach.
- Integration and features based code structuring of different components of the systems.

Basic rules of refinement chart

- Parallel Refinement
- Sequential Refinement
- Nested Refinement

$$M1 \sqsubseteq (M2 \parallel M3 \parallel \ldots\ldots \parallel M_{n-1} \parallel M_n)$$
$$M1 \sqsubseteq (M2 \succ M3 \succ \ldots\ldots \succ M_{n-1} \succ M_n)$$
$$(M1 \sqsubseteq M2,\ M2 \sqsubseteq M3, \ldots\ldots,\ M_{n-1} \sqsubseteq M_n)$$



$$M1 \sqsubseteq (M2 \parallel M3)$$
$$M1 \sqsubseteq (M2 \succ M3)$$
$$(M1 \sqsubseteq M2,\ M2 \sqsubseteq M3)$$

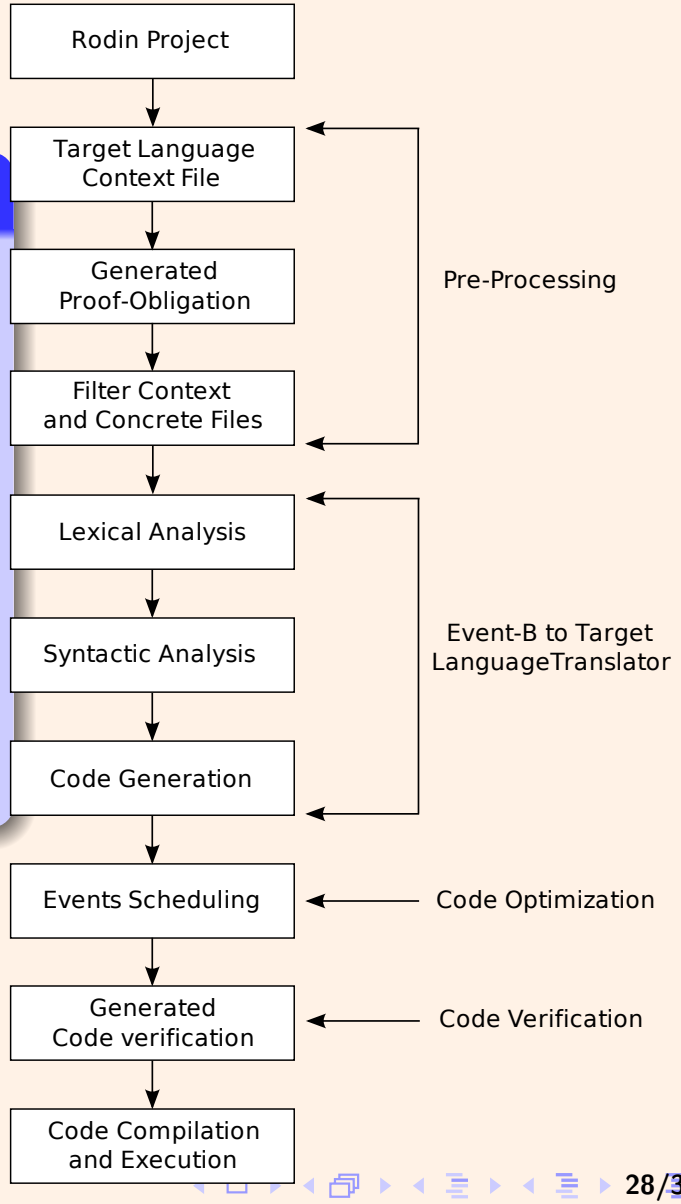
EB2ALL Code Generation Tool

EB2ALL

- Automatic code generation from Event-B formal specification to multiple languages (C, C++, Java, C#) to make it executable.
- A module for Rodin (Deploy is an European Commission FP7 project)
- It is more than 20000 lines of code
- More than 1000 downloads

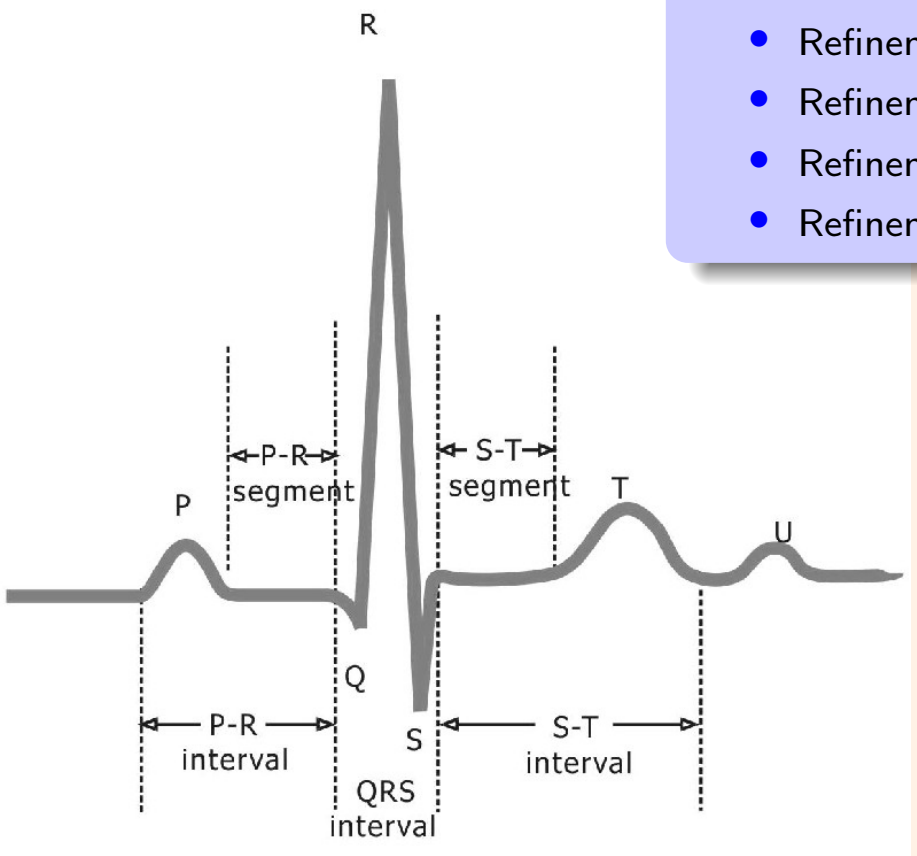
Download : http://eb2all.loria.fr/

EB2ALL Tool Architecture



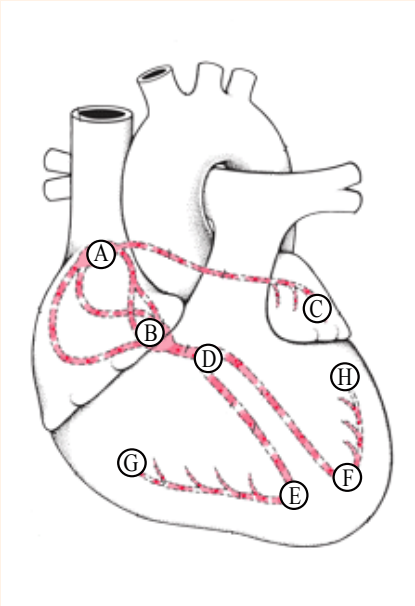
Formal Logic Based Heart-Model

Electrocardiogram (ECG)



Formal Development

- Abstract Model
- Refinement 1 : Introducing Steps in the Propagation
- Refinement 2 : Impulse Propagation
- Refinement 3 : Perturbation the Conduction
- Refinement 4 : Getting a Cellular Model



Landmarks on the Electrical Conduction of Heart

Current Summary

1 Verification

2 Proof Management

3 Development Methodology and Tools for Correct By Construction

4 Project Management

Steps of the project

- Reading the description of the problem to solve.
- List the safety properties as partial correctness, mutual exclusion, ... and properties on sets, constants, ...
- Identify the state variables for the first machine
- Define inductive definitions if necessary and state correctness properties as $\forall n.n\mathbb{N} \Rightarrow n * n = v(n)$.
- List Use Cases : for instance, adding a new user, updating a a phone number, recording a movement ...
- First expression of a context and a machine.

Documentation

- **Property**
  - *the set of people is non empty and finite*
  - $people \neq \varnothing \wedge finute(people)$