ClearSy

BART – Automatic Refinement in B
Introduction

- **Typical B development**
  - Abstract model: describes the requirements of the software
  - Concrete model: refinement of the abstract model, describing the implementation

- **Aims of automatic refinement**
  - Automatically generate the concrete model (or part of it) from the abstract model
Example project

Abstract model

Concrete model

Automatically generated

B component
B specification
Refinement or implementation
Why use automatic refinement?

- **Productivity**
  - Security Critical Software usually requires twice more workload
  - Automatic refinement can divide the workload by two
    - Automatic generation of the concrete model
    - Model easier to validate
  - CSC software developed with non CSC budget

- **Allows focusing the work on the “interesting” parts of the development**
  - Writing the specification, optimization of the software, etc...
1997: Matra Transport International (Siemens Transportations Systems) develops internally a tool called edithB still in use in the company
  ▪ 1999: FM'99 Workshop: Automatic refinement

2008: ClearSy develops BART in order to allow the community to benefit from automatic refinement tools
  ▪ 2009: Atelier B 4.0 with integrated BART tool
Uses of automatic refinement

- Used worldwide for several metros

- Biggest implementation: Val de Roissy Shuttle
  - Alarm control unit: 265 kloc B model (40 kloc handwritten), 186 kloc Ada code
  - Section automatics pilots: 67 kloc B model, 50 kloc Ada code
The BART tool

- Automatic refinement tool integrated in Atelier B
  - Open-Source tool
- Uses refinement rules to refine the components
  - The user can add new rules on a project and component basis
    - In the case where no existing rule is provided
    - To implement more efficient refinement
- The generated components still have to be proved
  - Makes BART a non-critical tool
  - Simplifies adding rules
    - An incorrect rule will generate non-proveable code
Working principles

- Input: complete set-theoristic model of a software
- Output: refinements and implementations

Refinement engine: applying transformation rules

```plaintext
RULE assign_a bool_subset_b_c_11
  REFINES
    @a := bool (@b <: @c-@d)
  REFINEMENT
    @a := bool (@b <: @c & @b \ / \ @d = { })
END;

RULE assign_a bool_belong_b_c_16
  REFINES
    @a := bool (@b->@d : @c @e)
  REFINEMENT
    @a := bool (@b : @c & @d : @e)
END;
```
BART usage

- Handles concrete refinements only
  - Requires an abstract machine detailed enough
  - Does not replace proof or specification refinements
  - The input model should be deterministic

- Provides an automatic and an interactive mode
  - Automatic mode: allows automatically applying the rules
  - Interactive mode: used to add rules when the default rules do not succeed
    - Allows inspecting the different rules that are applied
    - After the rules have been added interactively, the automatic mode is used
Using BART in a project

- Using BART usually means writing refinement rules instead of refinements
  - Rules written using the interactive mode
  - Default set of rules usually not enough/suitable for large projects
  - Rules can be kept and reused for later projects

- Refinement rules have to be written
  - On a project basis (rare)
  - For specific components (often)
Writing refinement rules

- Adding rules on a project basis
  - In the case where the specification is written in a different way than expected from the default rules
  - Typically an expert activity

- Adding rule for specific components
  - When the default rules cannot refine the component
  - When the default refinement is not efficient enough
    - Adding optimisations
    - Using a better refinement scheme
Refinement rules

- Refinement rules applied by pattern-matching

- Three main types of rules
  - Variable rules, for data refinement
  - Operation rules, for algorithmic refinement
  - Initialisation rules, specific case of algorithmic refinement

- Rules contains
  - A pattern, indicating the kind of element that can be refined
  - Guards, that indicates whether a rule can be applied or not
  - A refinement pattern, that indicates the resulting refinement
Conclusion

Automatic refinement can improve productivity by
- Automatizing repetitive tasks
- Leading to simpler proofs
- Simplifying reusing of known refinement patterns

BART
- Integrated in Atelier B 4.0
- Open-Source Tool
- Supported by grant No ANR-06-SETI-015-03 awarded by “Agence Nationale de la Recherche”
- Part of RIMEL project: “Incremental refinement of event models”
Demo

- Refinement of a small project with BART
12 mars 2010

Thanks