Constructing Correct Communicating Systems

Distributed Refinement in the B Method for CORBA -MATISSE Project-

Traian MUNTEAN
Univ. “Méditerranée”-Marseille
CNRS-TIMA Grenoble
muntean@imag.fr

Acknowledgements: O. ROLLAND & C. ENE
Motivation

Evolution of Programming Models

- 50s: machine language (sequential) programming
- 60s: procedure-based (sequential) programming
- 70s: structured programming
- 80s: objects/actors oriented (concurrent) programming
- 90s: distributed object-based (parallel) programming

- 00s THESIS:
  - interconnection/interaction-based programming
Methodology

Constructing vs. Guaranteeing “quality” of systems

Evaluating experimental/generic systems

Increasing efficiency for system design

Proved evolution/scalability of a system

Predictable faults

Methods of refinement for construction of correct communicating systems
Refinement

Transforming an abstract model into a concrete one

- To add details from the abstract specification
- To reduce indeterminancy
- To get correct (through formal proofs) implementations
- Adaptive, optimal systems
- Reduce a posteriori testing and/or proofs (model checking)
- e.g. B-Method, Action Systems
Distributed refinement (1)

Specification

Refinement

\[ P_1 \parallel P_2 \parallel \ldots \parallel P_n \]
## Distributed refinement (2)

<table>
<thead>
<tr>
<th>Distribution through stepwise refinements</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗ Refinement</td>
</tr>
<tr>
<td>✓ To distribute a monolithic specification</td>
</tr>
<tr>
<td>✓ To automatically generate communications through refinement</td>
</tr>
<tr>
<td>✓ To refine communications separately from operations</td>
</tr>
<tr>
<td>✗ Create new processes</td>
</tr>
<tr>
<td>✓ To integrate details from the specification</td>
</tr>
<tr>
<td>✓ To use abstraction / modularity at any level of spec</td>
</tr>
<tr>
<td>✗ Abstraction-Grouping</td>
</tr>
<tr>
<td>✓ To sequentialize processes</td>
</tr>
<tr>
<td>✓ To map/adapt to the target topology</td>
</tr>
<tr>
<td>✓ Optimisation - Embedding</td>
</tr>
</tbody>
</table>
Distributed refinement (3)

System

\[ \text{Operations} \]
\[ \begin{align*}
\text{ev}_1 &= g_1 \Rightarrow P_1 \\
\text{ev}_2 &= g_2 \Rightarrow P_2 \\
& \quad \vdots \\
\text{ev}_n &= g_n \Rightarrow P_n
\end{align*} \]

Refinement

\[ \text{Operations} \]
\[ \begin{align*}
\text{ev}'_1 &= g'_1 \Rightarrow P'_1 \\
\text{ev}'_2 &= g'_2 \Rightarrow P'_2 \\
& \quad \vdots \\
\text{ev}'_m &= g'_m \Rightarrow P'_m
\end{align*} \]

if and only if

\[ (\text{ev}_1 \ldots \text{ev}_n)^\prec \prec (\text{ev}'_1 \ldots \text{ev}'_m)^\prec \]

or

\[ (p_1 \ldots p_n)^\prec \prec (p'_1 \ldots p'_m)^\prec \]
## Distributed refinement (4)

### Operators
- **Specification operators**
  - Creation of processes
  - Abstracting processes (factorization)
  - Unbounded distribution of processes

- **Implementation operators**
  - Sequencialization of processes
  - Abstraction / parallelization of loops
  - Abstraction / parallelization of conditions

---

“Distributed Refinement using the B Method” O. Roland & T. Muntean
Matisse Project Report, 2000
Example: ABR protocol

Available Bit Rate for ATM switches

\[ v_1 \quad v_{j-1} \quad v_j \quad v_k \quad v_{k+1} \quad v_l \]

- \( t - t_2 \)
- \( t - t_3 \)
- \( t \)

dead zone  sensitive zone  advanced zone

\( t_2 \): delay after which values are meaningless

\( t_3 \): delay after which values are meaningful

**ABR algorithm dynamically computes the highest possible dataflow rate**

\[ A \geq \max (v_j \ldots v_k) \]
**ABR: Specification in B**

**Invariant**

\[ l \in \mathbb{N} \]
\[ v \in (1..l) \rightarrow \mathbb{N} \]
\[ j \in 1..l \]
\[ k \in j..l \]
\[ A \in \mathbb{N} \]
\[ \max (v[j..k]) \leq A \]

**Initialization**

\[ j = k = l = 1 \]
\[ A = v(1) \]

**ext =**

any \( X \) where

\( X \in \mathbb{N} \)

then

\[ l, v(l + 1) := l + 1, X \]

end

**in =**

select \( k < l \) then

\[ A : (\max(v[j..k+1]) \leq A) \land \]
\[ k := k + 1 \]

end

**out =**

select \( j < k \) then

\[ A : (\max(v[j+1..k]) \leq A) \land \]
\[ j := j + 1 \]

end

**inout =**

select \( j < k \land k < l \) then

\[ A : (\max(v[j+1..k+1]) \leq A) \land \]
\[ j, k := j + 1, k + 1 \]

end
**ABR: First refinements (iterating)**

\[
in1 =
\begin{align*}
&\text{select} \\
&\quad k < l \land \max(v[j..k+1]) \leq A' \\
&\quad \text{then} \\
&\quad A := A' || \\
&\quad k := k + 1 \\
&\text{end}
\end{align*}
\]

\[
in2 =
\begin{align*}
&\text{select} \\
&\quad k < l \land A' < \max(v[j..k+1]) \\
&\quad \text{then} \\
&\quad A' \in N \\
&\text{end}
\end{align*}
\]

\[
in1 =
\begin{align*}
&\text{select} \\
&\quad k < l \land v(k+1) \leq A \\
&\quad \text{then} \\
&\quad k := k + 1 \\
&\text{end}
\end{align*}
\]

\[
in2 =
\begin{align*}
&\text{select} \\
&\quad k < l \land A < v(k+1) \\
&\quad \text{then} \\
&\quad A := v(k+1) \\
&\text{end}
\end{align*}
\]
ext =
  any X where
  x ∈ N
  then
  l, L := l + 1, X  ||
  choice
    select k < f ∧ F < X then
      F := X
    end
  or
    select f ≤ k ∧ A < X then
      f, F := l + 1, X
    end
  end
end
**ABR: Intermediate (selecting)**

**ext1** =
any
  \[ X \in \mathbb{N} \land k < f \land F < X \]
then
  \( l, L, F := l + 1, X, X \)
end

**ext2** =
any
  \[ X \in \mathbb{N} \land k < f \land X \leq F \]
then
  \( l, L := l + 1, X \)
end

**ext3** =
any
  \[ X \in \mathbb{N} \land f \leq k \land A < X \]
then
  \( l, L, f, F := l + 1, X, l + 1, X \)
end

**ext4** =
any
  \[ X \in \mathbb{N} \land f \leq k \land X \leq A \]
then
  \( l, L := l + 1, X \)
end
**ABR: Last refinement(1)**

\[
\text{ext1} = \\
\text{any} \\
X \in N \land X \leq F \land T < TF \\
\text{then} \\
F, L, TL := X, X, T + t2 \\
\text{end}
\]

\[
\text{ext2} = \\
\text{any} \\
X \in N \land X \leq F \land X \leq T \land TL \neq T + t2 \\
\text{then} \\
L, TL := X, T + t2 \\
\text{end}
\]

\[
\text{ext3} = \\
\text{any} \\
X \in N \land X \leq F \land X \leq T \land A < X \land TL \neq T + t2 \\
\text{then} \\
F, L, TL, TF := X, X, T + t2, T + t3 \\
\text{end}
\]

\[
\text{ext4} = \\
\text{any} \\
X \in N \land X \leq F \land X \leq T \land A < X \land TL \neq T + t2 \\
\text{then} \\
L, TL := X, T + t2 \\
\text{end}
\]

\[
\text{ext} = \text{ext1} || \text{ext2} || \text{ext3} || \text{ext4}
\]
abr =
  any X where
  x ∈ N ∧ TL ≠ T + t2
  then
  L, TL := X, T + t2 ||
  if T < TF then
    if F < X then
      F := X
    end
  else
    if A < X then
      F, TF := X, T + t3
    end
  end
end
end
scheduler1 =
select
    T + 1 \neq TF \land T + 1 \neq TL
then
    T := T + 1
end

scheduler2 =
select
    T + 1 = TF
then
    T := T + 1 \|
    A := F
end

scheduler3 =
select
    T + 1 = TL
then
    T := T + 1 \|
    A := L
end
ABR: Last refinement (conditioning)

scheduler =
    begin
    T := T + 1 ||
    if T + 1 = TF then
        A := F
    else
        if T + 1 = TL then
            A := L
        end
    end
end
end

Result: scheduler || abr

correctly implements ABR by construction
Communications – Broadcasting Abstract Machines

- Extending B Method for automatic generation of parallel broadcasting AMs
Point-to-point vs. Broadcast Formal Models

- process calculi based on point-to-point communications
  (ACP, CCS, $\Pi$-calculus, etc.)

- a few models based on multi-way synchronization or broadcast
  (CSP, SCCS, I/O automata)

- or mobility
  ($\Pi$-calculus, CHOCS, HO$\Pi$-calculus)

- models based on broadcast which can express mobility?
  (recently: HOBS, dynamic I/O automata)

"b$\Pi$-calculus" Ene & Muntean FCT'99 (LNCS), FMPPTA'01(IEEE)
Co-inductive refinement

- $b\Pi$ (Broadcasting Mobile Systems)
  - Process algebra \textit{(dynamically evolving systems of mobile processes)}
  - Communications by synchronous broadcast
  - Co-inductive reasoning

- Simulation
  - Co-inductive relation
  - Appropriate for infinite systems
  - Refinement

- Translation from B abstract machines to $b\Pi$ processes
  - Implementing shared memory systems into message passing systems
  - Co-inductive refinement operator

\textit{Ene\&Muntean « A Broadcast-based Calculus for Mobile Communicating Systems », IEEE-FMPPTA’01}
Behavioral equivalencies

- Trace equivalence
- Bisimulations (weak, barbed, strong)
- Testing equivalence
**Encoding**

- **Question**: *Can we uniformly implement broadcast communications using "just" point-to-point communications?*

- **Theorem**: *There exists no uniform encoding of the $b\Pi$-calculus into $\Pi$-calculus*

  C. Ene, T. Muntean FTC’99; Springer-Verlag LNCS
  “On the expressive power of point-to-point and broadcast communications”

**Results:**
- A broadcast calculus with dynamic scoping
- Separability results between calculi based on point-to-point communications and broadcast communications
- Behavioral equivalencies for $b\Pi$-calculus
- Full axiomatization for the above equivalences
Refinements of given Specifications towards correct (proved) Implementations

- $Spec \ «_{must} Impl$
  
  (whenever $Spec$ is correct, $Impl$ is correct)

- $Impl \ «_{may} Spec$
  
  (if $Impl$ is faulty in a given environment, so is for $Spec$ in this environment)
Communicating Machines through standard communication interfaces

- Communication model
  - Message passing through channels
  - Shared memory
  - Events
  - Mobility & Diffusion
  - RPC / RMI

- Synchronization model
  - Synchronous
  - Asynchronous

- Transmission model
  - Typed data
  - Untyped data
Communications – B refines for CORBA

- Extending B Tool for automatic generation of communication interfaces for ORBs
CORBA (1)

Client

Hidden operations

IDL Stub

Object (service)

Exposed operations

Hidden operations

IDL Skeleton

Request

Object Request Broker
The exposed_service machine contains all the necessary information to generate the IDL.

It contains also behavioural and operational information of the functions described in the IDL (more semantics than in classical IDL).
New refinement strategy for B tool

- Initial Spec. $M_0$
- Refined $M_i$

Distributed, Standard or Event B Refinement

Decompose (Application Oriented)

Client$_i$, Interface$_i$, Server$_i$
Conclusions

- **Distributed refinement method**
  - Time / Communication stretching and folding

- **Models of communication**
  - Communications
    - correct by construction
    - automatically generated
  - Translation into broadcasting virtual machines
    - Co-inductive refinement of abstract machines
    - correctness of implementations (i.e. routing)
  - Automatic generation of CORBA IDL and associated code

**Correct behaviours vs. non-faulty behaviours**