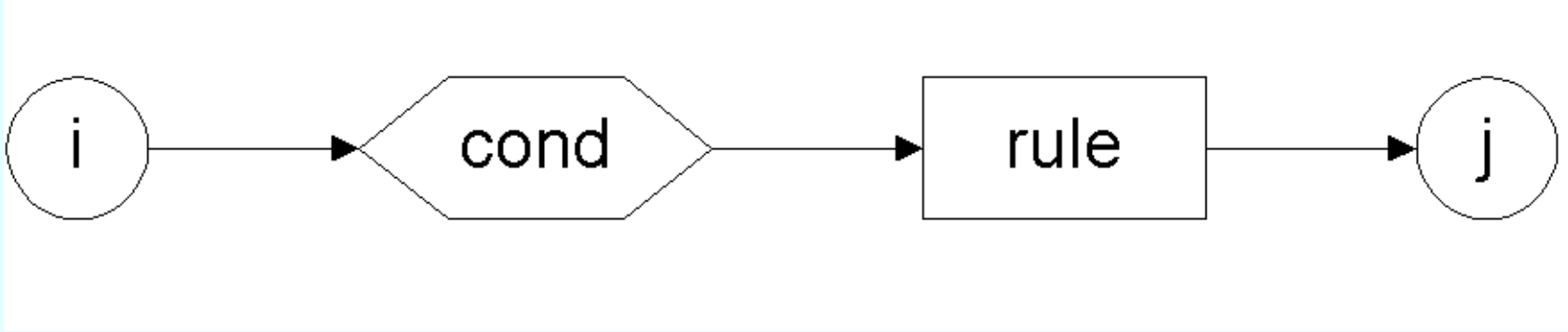
Egon Börger (Pisa)

Relating Event-B and ASMs

Università di Pisa, Dipartimento di Informatica, I-56127 Pisa, Italy
boerger@di.unipi.it

ASMs = FSMs with Abstract States



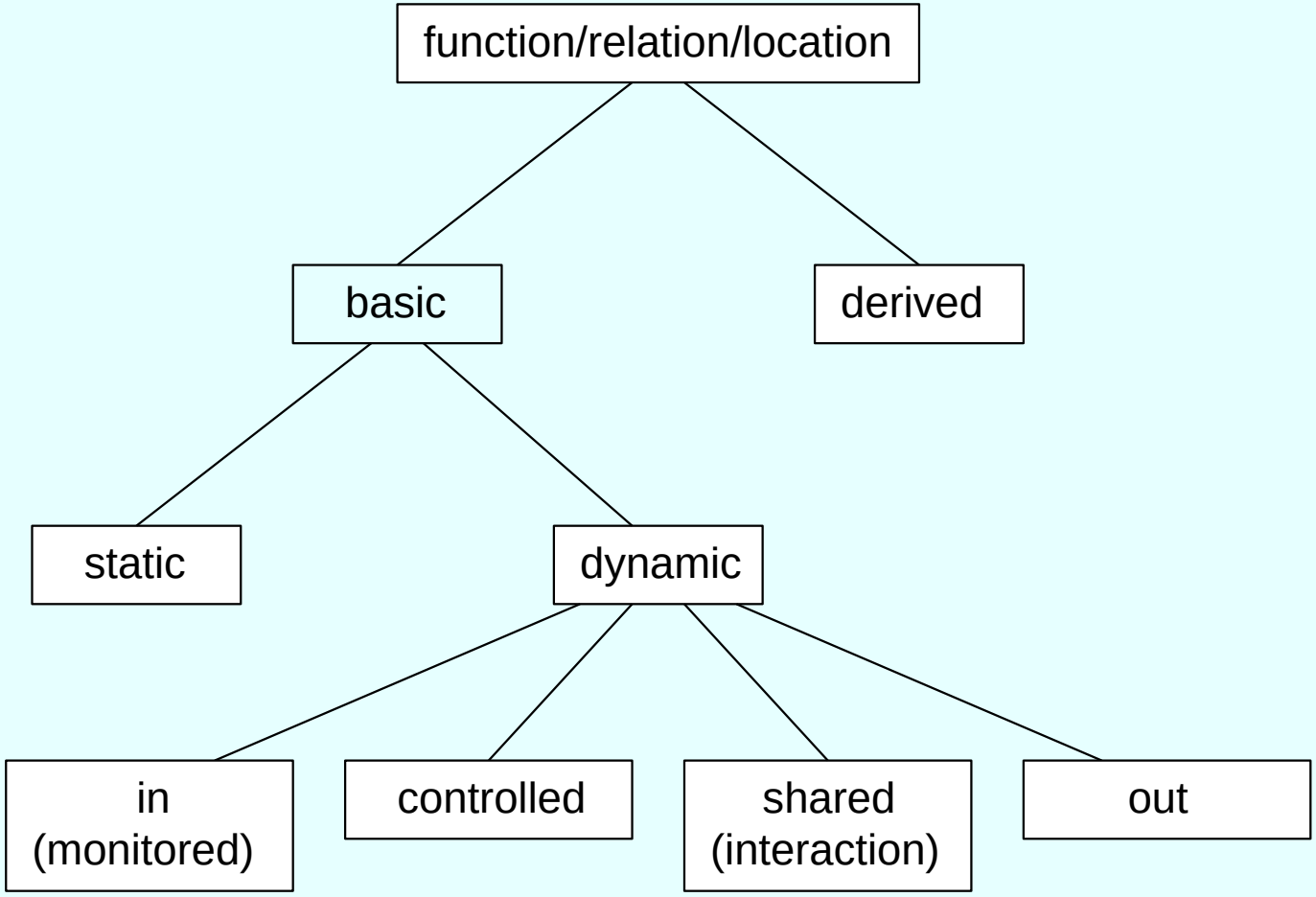
if $ctl\_state = i$ **then**
  if $cond$ **then**
    $rule$
    $ctl\_state := j$

where $cond \equiv input = a$ $rule \equiv output := b$ for FSM

ASMs use *parameterized locations* and *first-order conditions*:

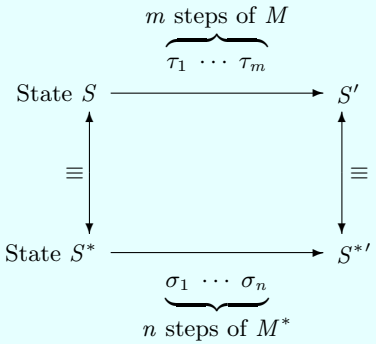
- rule = set of updates $f(t_1, \ldots, t_n) := t$
  allowing to **choose** among or to generalize **forall** params
- cond = arbitrary first-order formula

Classification of ASM Locations/Functions



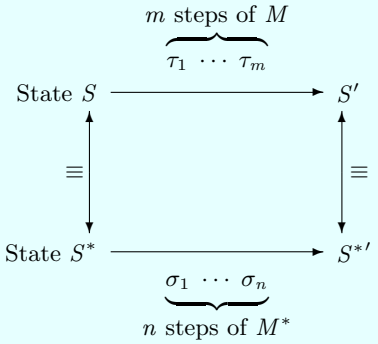
supporting the separation of concerns: information hiding, data abstraction, modularization and stepwise refinement

ASM Refinements and their Parameters



- a notion of *refined state*
- a notion of *states of interest* and of *correspondence* between M-states S and M^*-states S^* of interest, e.g. initial/final states (if present)
- a notion of *computation segments*
  - abstract $\tau_1, \ldots, \tau_m$ of single M-steps τ_i
  - corresponding refined $\sigma_1, \ldots, \sigma_n$ of single M^*-steps σ_j

ASM Refinements and their Parameters (Cont'd)



- a notion of *locations of interest* and of *corresponding locations*, i.e. pairs of sets of locations one wants to relate in corresponding states
- a notion of *equivalence* $\equiv$ of the data in the locations of interest

In (m, n)-diagrams or (m, n)-refinements, $\tau_1, \ldots, \tau_m$ and $\sigma_1, \ldots, \sigma_n$ lead from corresponding states of interest to (usually the next) corresponding states of interest

Definition of Correct ASM Refinement Step

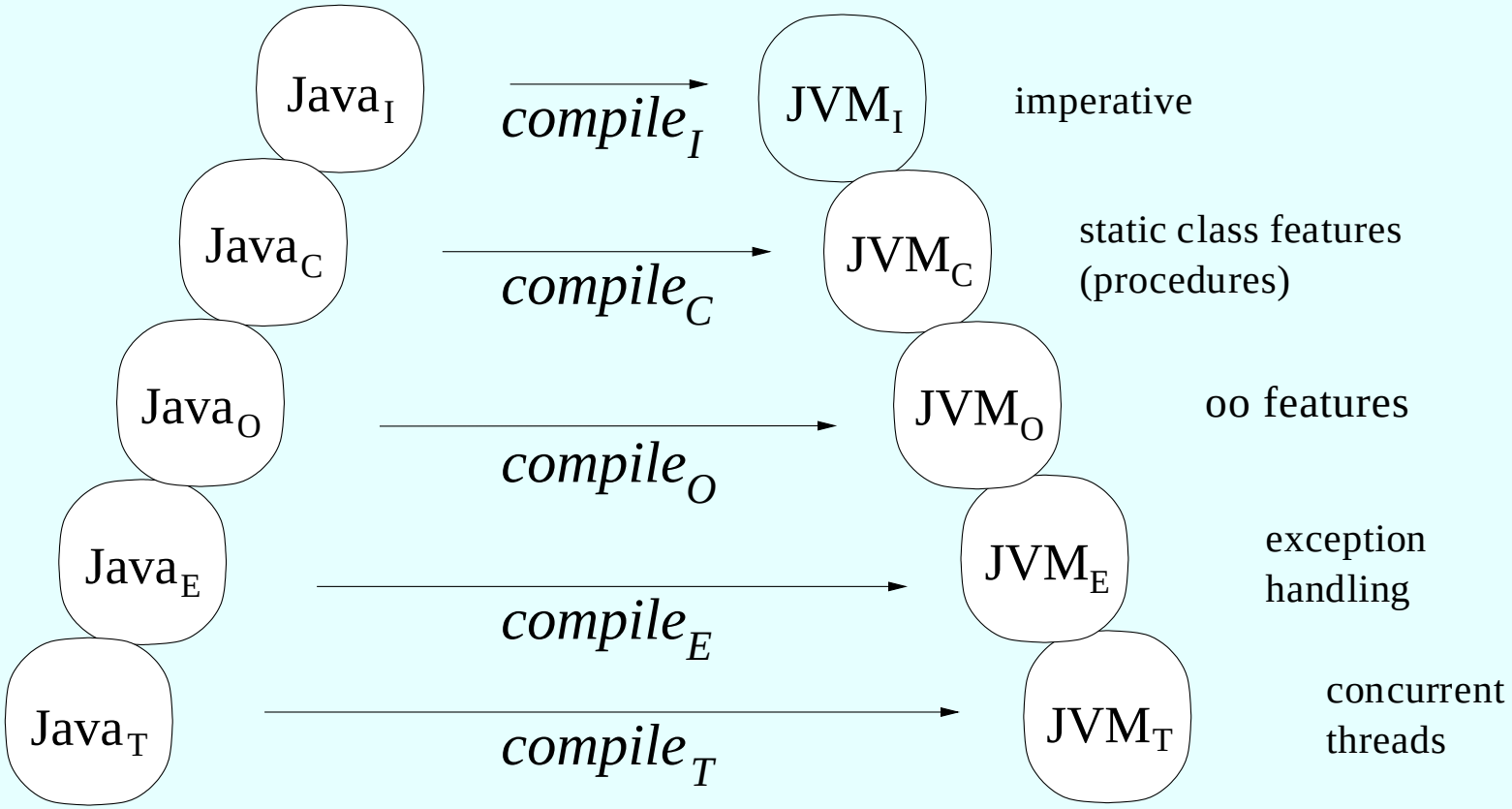
Fix any notions $\equiv$ of equivalence of states and of initial and final states. M^* is called a *correct refinement* of M if and only if for each M^*-run $S_0^*, S_1^*, \ldots$ there are an M-run $S_0, S_1, \ldots$ and sequences $i_0 < i_1 < \ldots, j_0 < j_1 < \ldots$ such that $i_0 = j_0 = 0$ and $S_{i_k} \equiv S_{j_k}^*$ for each k and either

- both runs terminate, their final states are equivalent, or
- both runs and both sequences $i_0 < i_1 < \ldots$, $j_0 < j_1 < \ldots$ are infinite

M^*-run $S_0^*, S_1^*, \ldots$ is said to simulate the M-run $S_0, S_1, \ldots$, where $S_{i_k}, S_{j_k}^*$ are the corresponding states of interest

- in (m, n)-refinements m, n may dynamically depend on states
- (m, n)-refinements with $n > 1$ or with $(m, 0), (0, n)$-steps support the feasibility of decomposing complex (global) actions into simpler (locally describable) ones
- procedural $(1, n)$-refinements with $n > 1$ have their typical use in compiler verification

Language-Oriented Refinement of Java/JVM into Layers



Layers are *conservative extension*s of each other and thus support componentwise design and analysis (validation & verification). Combination with an appropriate parameterization provides an *orthogonal treatment of language constructs* ("instructionwise").

Event-B Models as ASMs: states, events, invariants

- *States*: structures of given signature with
  - static part ('context')
    - sets s ('universes'), constants c, properties(c,s) ('axioms')
  - dynamic part: variables v and env (viewed as another model)
    - 'Inputting is done by non-determinacy'
  - initialization (via a special event with guard true)
- *Events* of form *If guard Then action*
  - guard: closed first-order set theory formula with $=$
  - action: one of the three forms
    - Updates: simultaneous substitution $v_1, \ldots, v_n := e_1, \ldots, e_n(x, v)$
    - **skip**
    - **choose** x **with** $P(x, v)$ **in** $Updates$
- *Invariant*: property holding in every state that is reachable from an initial state

Rule Normal Form in Event-B Models

- Normal form $Rule_1$ **or** $\ldots$ **or** $Rule_n$ with $Rule_i$ of form
  - **if** $cond$ **then choose** x **with** $P(x, v)$ **in** $Updates$
    - possible cases: P = true or Updates = $\emptyset$(**skip**)
- Def. R **or** $S = (\textbf{choose } X \in \{R, S\} \textbf{ in } X)$
- constraints for events:
  - no two events can occur simultaneously. Also distributed event-B models are based on this interleaving view
  - splitting into events implies some non-deterministic scheduling of events with overlapping guards
- constraints for rules:
  - no parallel update allowed for the same variable
  - no rules of form: **forall** x **with** $P(x)$ **do** $rule$
  - external choose only on rules (interleaving model), no further nesting of choices allowed in Updates

Event-B Refinement Notion

- only $(1, n)$ refinements with $n > 0$ satisfying:
  - in a refinement $F_1 \ldots F_n, F$ of E, each F_i refines $\mathbf{skip}$
  - the new events F_i do not diverge
  - if the refinement deadlocks, then the abstraction deadlocks
- variables in abstract and refined model are pairwise different
- no $(1, 0)$ refinement ('each abstract event must be refined by at least one refined event')
- no (n, m) refinement with $n \neq 1$
- observables = locations of interest ('projections of state variables')
  - fresh (for state vars and for invariants)
  - modifiable only by 'observer event' $a := A(v)$
  - dependent only on state variables v
  - abstract observables $A(v)$ can be 'reconstructed' from refined ones by an equation $A(v) = L(B(w))$ ('invariant gluing the abstract observables to the refined ones')

References

J.-R.Abrial: *Discrete System Models* Manuscript, September 2004 (Version 2)

J.-R.Abrial: *Event Driven Distributed Program Construction* Manuscript, August 2004 (Version 6)

E. Börger: *The ASM Refinement Method* Formal Aspects of Computing 15 (2003), 237-257

G. Schellhorn: *Verification of ASM Refinements Using Generalized Forward Simulation* JUCS 7.11 (2001) 952–979

G. Schellhorn: *ASM Refinement and Generalizations of Forward Simulation in Data Refinement: A Comparison* TCS 336/2-3 (2005) 403-436

E. Börger and R. F. Stärk: *Abstract State Machines* Springer 2003. pp.X+438. See *http://www.di.unipi.it/AsmBook/*