VII Möbius —— Reza Pulungan.

Notiztitel 16.03.2005

VIII. StoCharts.

A] UML: Unified modelling language.

- a family of languages (mostly graphical)
- to describe software systems.
- standardised by the OMG (Object Mangement Group)
- Examples:
  - class diagrams / static structure diagram
  - sequence diagrams (message sequence charts)
  - statechart diagrams

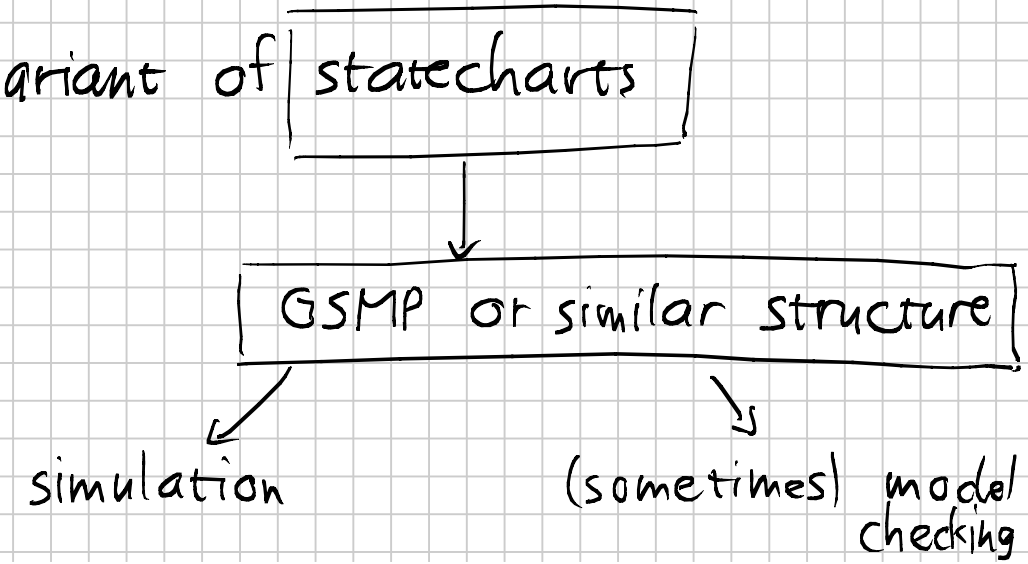
How to use the UML?

1. Sketches: write down first informal ideas.
   No strict syntax, no clear semantics.

2. Blueprint: overview
   simple models, clear syntax & semantics
   not too slow — formal analysis

3. Graphical programming language.
   people start to hack.

Our idea: use a variant of **statecharts**

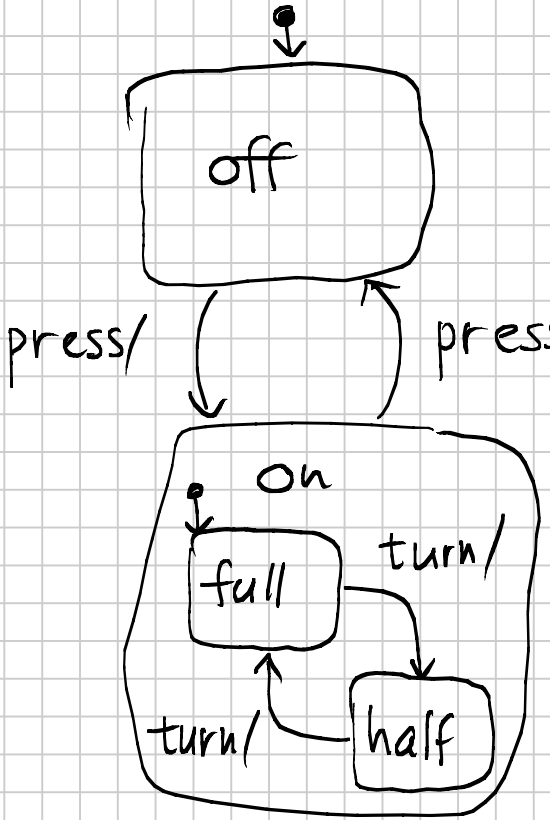


B Statecharts.

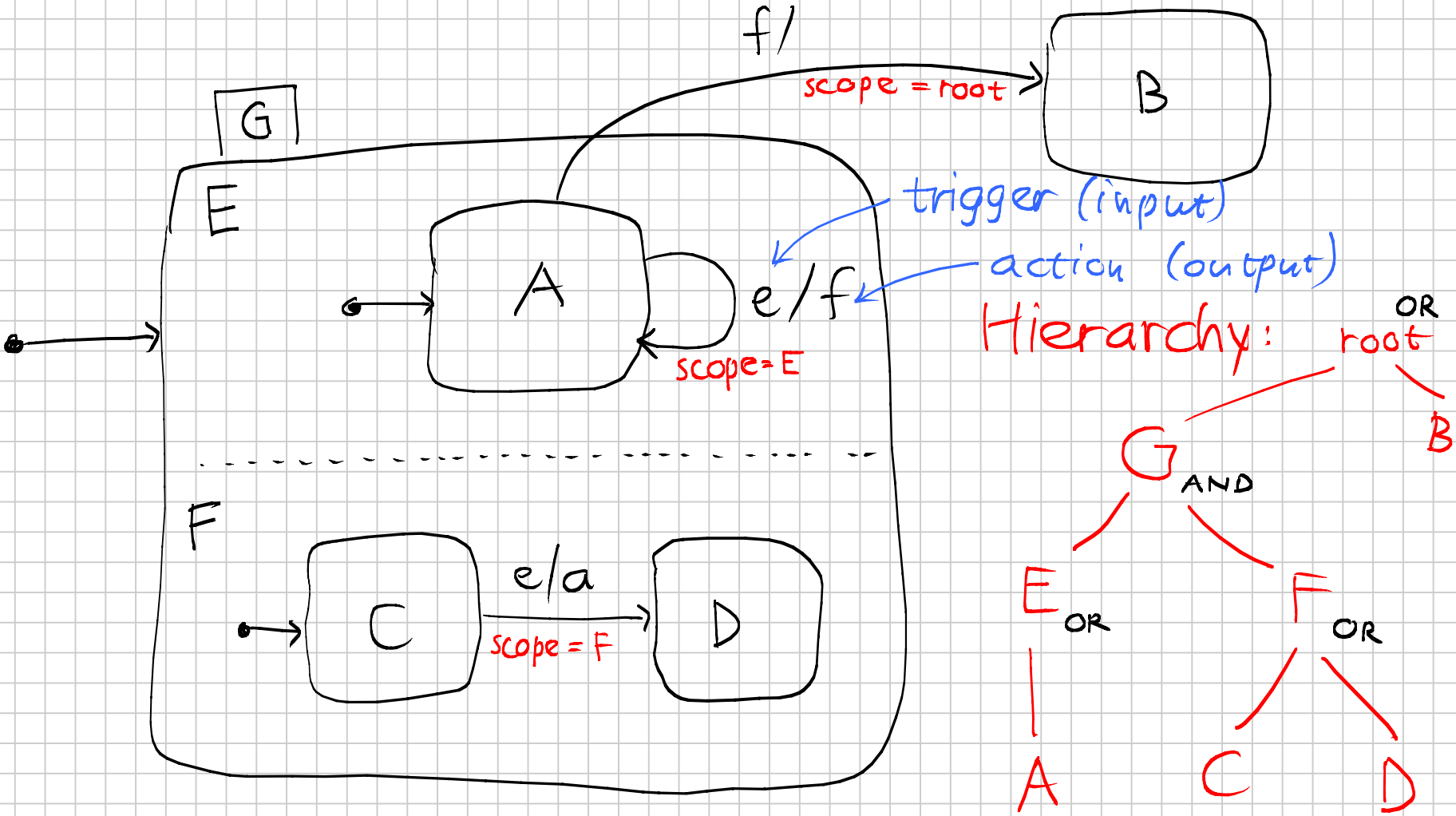
Finite automata + hierarchy + parallelism

states can be refined into substates.

states may contain multiple parallel processes/substates



Hierarchy: state "on" is refined into "on.full" and "on.half".



Hierarchy:

Parallelism: indicated by ..... . E and F are parallel.

If the system is in state G, it also is in states E and F.

Abstract syntax. A statechart consists of:

- a set N of nodes, with a tree structure.
  (with a root node, not drawn)

  a function type: $N \longrightarrow \{$ AND, OR, BASIC$\}$

  type (root) = OR. type (leaf) = BASIC.

  a function default: $N \rightarrow\!\!\!\!\!\!+\; N$ that assigns a default node to each OR node.
- a set E of events
- a set G of guards
- a set A of actions
- a set Edge of edges. Edges are quintuples

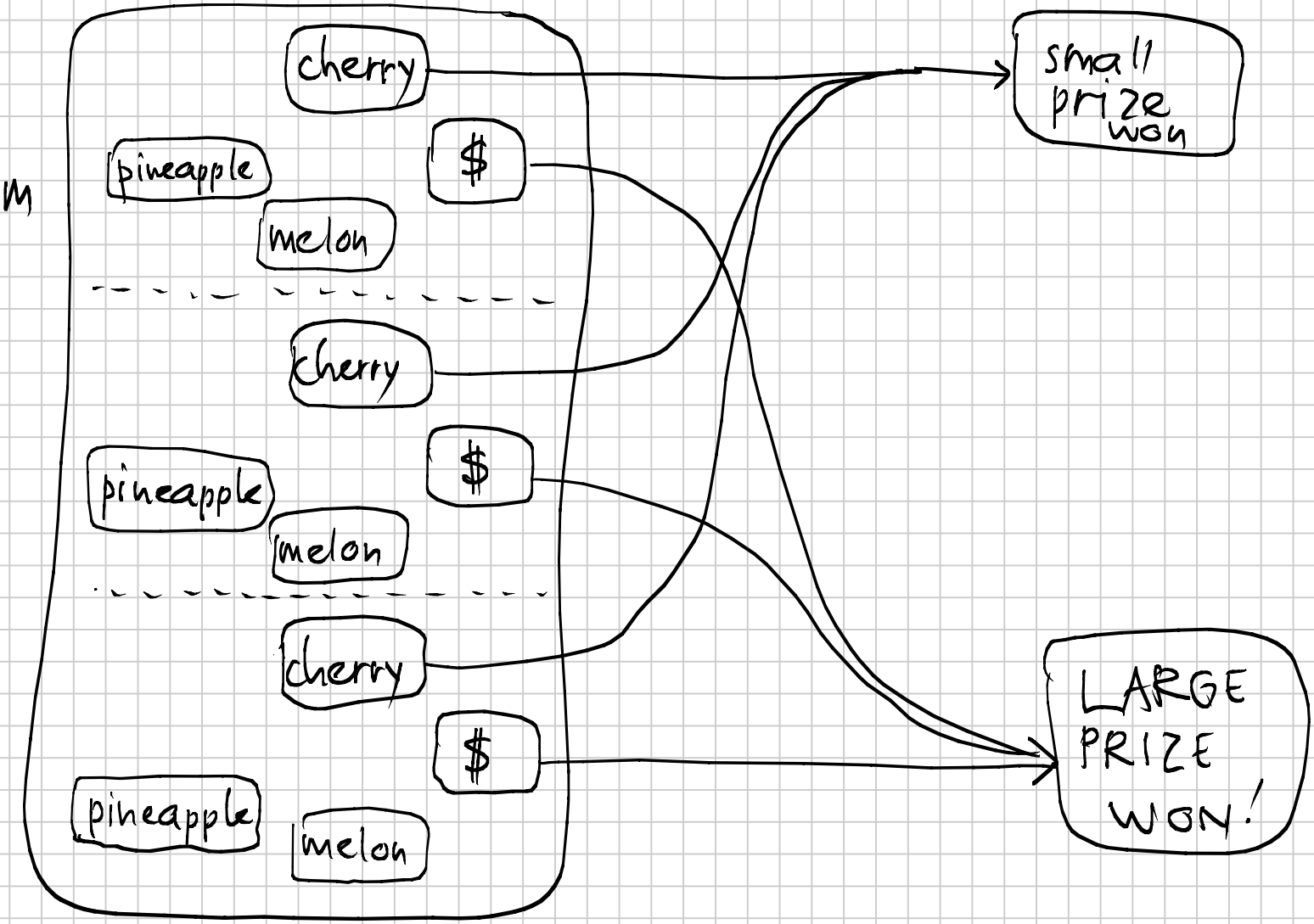
  (X, e, g, A, Y) $\quad \emptyset \neq X \subseteq N \quad e \in E \quad g \in G$

  $A \subseteq A \quad \emptyset \neq Y \subseteq N$

  X: source nodes Y: target nodes.

Example of multiple source nodes.

a gambling machine with three reels.



Parallelism helps to keep this drawing simple.

(draw 15 states instead of 64)

Statechart semantics.

Configuration: $C \subseteq N$ that satisfies the conditions

1. root $\in C$.
2. $n \in C$, type(n) = OR $\Rightarrow \exists!$ child c of n: $c \in C$
3. $n \in C$, type(n) = AND $\Rightarrow \forall$ child c of n: $c \in C$

State = (configuration, set of events) = (C, I)

events received but not yet processed.

An edge (X, e, g, A, Y) is enabled if $X \subseteq C$ and $e \in I$ and g holds.

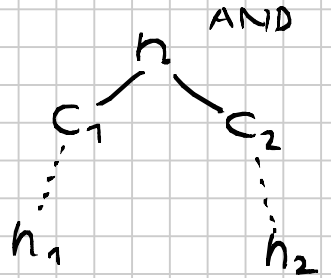
The scope of an edge is the least common ancestor of $X \cup Y$.

Two edges are **inconsistent** if their scopes overlap.

Two nodes n_1, n_2 are **orthogonal** if $\exists\, n \in N$, $type(n) = \text{AND}$ and $\exists$ children c_1, c_2 of n, $c_1 \neq c_2$, such that

c_1 is an ancestor of n_1 and

c_2 " " " " n_2



Two edges are **inconsistent** if their scopes are not orthogonal.

A **step** is a maximal set of enabled, pairwise consistent edges.

Step execution:

1. leave all nodes in the scope of some edge of the step
2. execute the actions of the edges in the step.
3. enter all target nodes of some edge in the step.
4. enter (implicitly) all other nodes, as necessary.