The Meadowview Language

Cheoljoo Jeong

Overview

The **Meadowview** is a programming language which can be used to define processes and interactions between processes.

In particular, the Meadowview language

- supports the notion of **agents** as containers of processes,
- supports the **global namespace** of *events* and *agents*
- supports the notion of **(local) real time** *within* agents,

Elements of Meadowview

An **agent**, which is unqiue in the entire runtime, is a container of processes.

An **event**, which is unique in the entire runtime, represents any "change of state" in which some processes may be interested in.

A **channel**, which is unique in the entire runtime, is a special type of agent which acts as a buffer for an event.

A **process**, which is unique within an agent, is a reactive code which can perform simple testing over event instance, generation of new event instance, execution of foreign program (e.g. Python script, Java program).

Agents

An **agent** has a name which is unique in the entire Meadowview system.

An agent, which is a *container of processes*, can send and/or receive events.

Agents are the only programming element which serve as the *destination of event delivery*.

Events

An **event** is a "change of state" which might be interesting to some agents (or processes).

An event is unique across the entire Meadow system and events cannot be generated or removed from inside a Meadowview program. They are *given* just like IP addresses in the Internet.

Though one cannot define events using the Meadowview language, the Meadow system provides API through which one can add/remove/update events. An event definition consists of a **name** and **datatype** definition. For example,

```
Event com.meadow.VacationApprovalRequest {
  com.meadow.EmployeeID id;
  Date date;
};
```

Event Instances

While an event represents a *class* of related state-changes, an an **event instance** represents an actual instantiation of an event. In the runtime (which is inevitably dynamic), event instances are transferred between agents.

Transfer of event instance from one device to another carries following information:

- notification that the given event has occurred
- payload associated with the event instance
- additional control information such as timestamp, sender, receiver

Processes

A **process** is a specification of behavior which will be **deployed** in an agent. A process consists of two parts: *event list* and *process body*.

The **event list** is an unordered list of events to which the process is sensitive to.

The **process body** is the code which is executed when any event in the event list occurs. Roughly, the code can perform following activities:

- *Testing over event instance*
- *Branching/looping based on the result of testing*
- *Terminal actions*:
  - send a new event instance
  - execute foreign code
  - delay for real-time (e.g. sleep(2 msec))

Processes: Control Constructs

Meadowview supports the following control constructs:

- **Sequencing**: ";" is a sequencing operator
- **Branching**: if-then-else
- **Loops**: for-loop, while-loop

Processes: Terminal Actions

There three types of **terminal actions** that a process can perform in reaction to events.

- Generate/send a new event instance to an agent.
- Execute a foreign program (e.g. Python script or Java program).
- Delay consumption.

Restriction on the type of terminal actions can result in a compact and efficient runtime.

Processes: Example

The following shows a simple process which models an approval of a vacation request.

```
process ManagerApproval @(com.meadow.ApprovalRequestFirst R) {
   Agent DB = Agent("com.meadow.DBEngine");
   reqhandle = SEND(DB, new com.meadow.DBQuery(R.employee, "manager");
   RECV(reqhandle, com.meadow.DBQueryResult mgr);
   reqhandle = SEND(DB, new com.meadow.Query(R.employee, "vac_used");
   RECV(reqhandle = com.meadow.DBQueryResult vacuse);
   if (vacuse.value() < 100 /* MAX_VAC */)
     SEND(mgr.agent(), new ApprovalRequestSecond(R));
   else {
     SEND(R.requester(), new ApprovalRequestDenied(NO_BALANCE));
}
```

Further improvements

- Must be able to designate an abstract event source: e.g. any `ApprovalRequestFirst` originated from an agent who is in "team member" relation with the agent where the `ManagerApproval` process is deployed.

Semantics of Meadowview Elements

Each (static) Meadowview element has its dynamic counterpart.

Syntax	Semantics
Agent	Device handle
Event	Event instance
Channel	Event queue
Process	Labeled transition system

Semantics: Events

Events exists to convey two pieces of information:

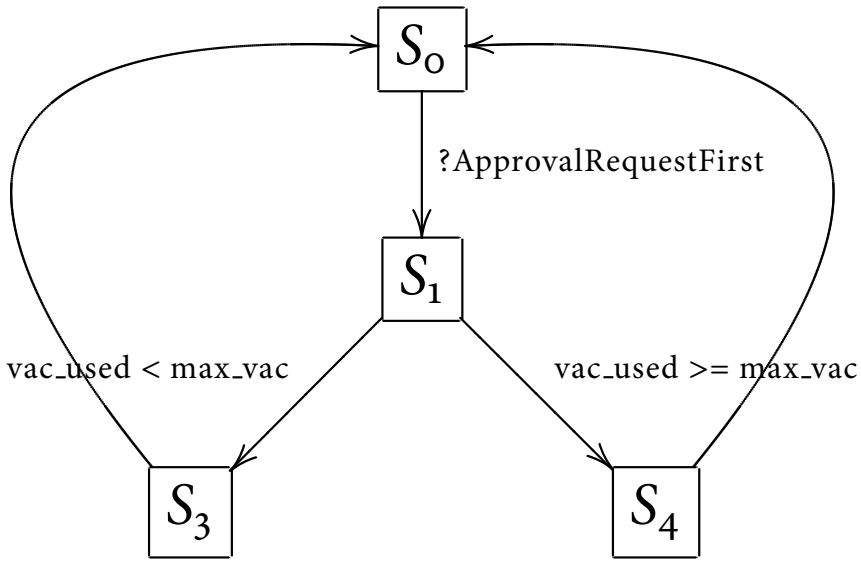
- notification that “something has happened”
- any associated data

An **event flow** is transfer of an event instance between two devices.

Event flows occur between devices not between processes. That is, the runtime running on a device is responsible for routing of event instances. The device runtime manages multiple processes in the given device.

Semantics: Processes

A process is compiled into a **labeled transition system** (or state machine with propositional formulas associated with edges).



An engine in an agent (device) is responsible for the execution of labeled transition systems.