DICE-R: Defining Human-Robot Interaction with Composite Events

Realizing multimodal human-robot interaction is complex

Multimodal interaction implies the combination of data from varying sources in sequence and in parallel—a complex problem, for which several dedicated solutions have been proposed.

Event-condition-action logic is used for end-user programming, but lacks scope management, support for combination of events.

State machines are frequently used to program robotics, but hidden for common use cases. State machines may not be ideal for end users to handle detailed task-level programming.

DICE-R specifies interactions per context using composite events

Context variables can be used to define relevance and variations of interactions.

Composite events determine the temporal constraints between the relevant events.

Event-Condition-Action rules link (combinations of) events to actions.

Example Multimodal Human-Robot Interaction

```
interaction pickAndGiveObject:
  always:
    camera.openHandAt<xh,yh,zh> + speech.test;
    camera.objectAt<name,xo,yo,zo>;
    robot.objectPicked<name>; camera.openHandAt<xm,ym,zm>*;
    speech.thanks - kinect.handClosed
  @detected camera.openHandAt<xh,yh,zh> + speech.test:
    speak 'going to fetch test'
  @timeout camera.openHandAt<xh,yh,zh> + speech.test:
    raise robot.logMessage<"failedStart">
  @timeout camera.openHandAt<xh,yh,zh>:
    speak 'show open hand to fetch object'
  @detect camera.objectAt<name,xo,yo,zo> when name = 'test':
    raise robot.pickObject<test',xo,yo,zo>
  @detect camera.openHandAt<xm,ym,zm>:
    raise robot.moveTo<xm,ym,zm>
  @end:
    raise robot.gripperOpen
  @detect kinect.handClosed
    when _lastEvent = robot.moveTo<xm,ym,zm>
    or _lastEvent = robot.objectPicked<name>:
      raise robot.returnObject
  @detect kinect.handClosed:
    speak 'canceled interaction'
```

DICE-R code shorter than equivalent textual Hasselt code

Less code duplication due to annotation of composite event

No need to know corresponding finite state machine