A taxonomy for triggered interactions using fair object semantics

Paul Gibson (NUI Maynooth)
Geoff Hamilton (Dublin City University)
Dominique Méry (Université Henri Poincaré Nancy 1 & LORIA & IUF)

18th of May 2000

FIW'00 at Glagow, Single Malt

thanks to CNET/FT/RD/PC, Enterprise Ireland, MAE$\oplus$CNRS, IUF

Overview of issues to be addressed

◇ Continuation of a work towards formulating a feature interaction algebra.

◇ Possibility to being able to perform a re-usable analysis of feature interactions based on feature classes.

◇ The weaknesses to this previous introductory work: informal classification hierarchies, the analysis that arose was based on a small number of very different features.

General Objectives

- ◇ Formalisation of the notion of triggered features
- ◇ Classification of a subset of common interactions.
- ◇ *Fair object* formal framework: integration of object-state machines and temporal logic.
- ◇ The development of a feature interaction algebra is not just a theoretical proposition but is also a practical engineering possibility.
- ◇ Testing our theoretical results in the construction of object oriented feature requirements models, where each feature is triggered by the same `dialIn` event.

Motivations

- ◇ the *application of formal methods* is fundamental to our approach.
- ◇ the *classification towards a taxonomy* corresponds to our re-usable analysis based on abstract classes.
- ◇ the *experimentation with different methods, architectures, platforms, etc ...* corresponds to the way in which we have developed our mixed semantic approach.
- ◇ Preliminary works: presentation at SNPD00 (Reims, Champagne)
- ◇ Applying the medicine to myself.

Fair objects: The need for fairness in requirements models

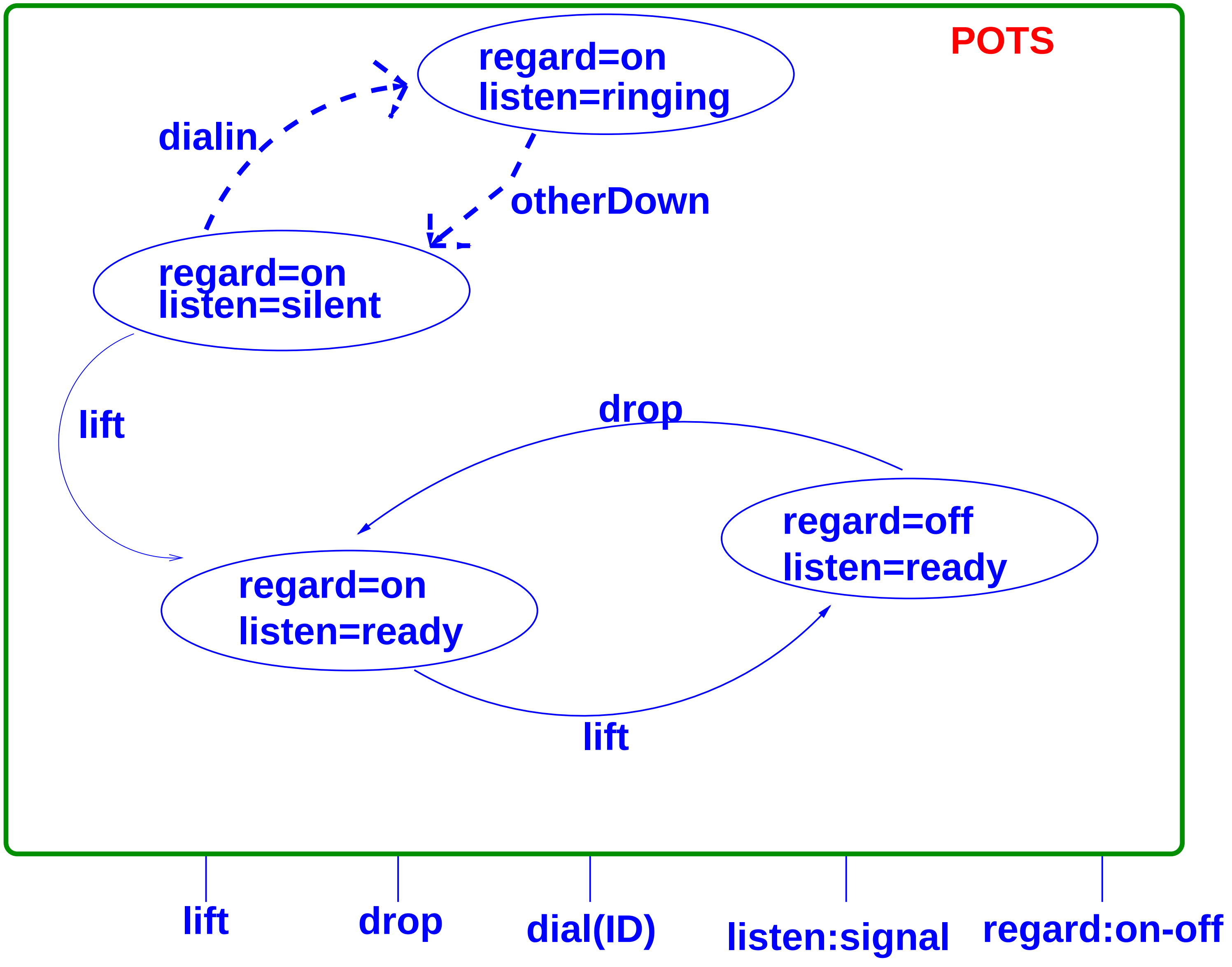
- ◇ An operational approach to requirements specification offers many advantages: improving communication between the customer and analyst, and aids in synthesis, analysis and validation.
- ◇ Inability to express *liveness* requirements: we can specify only what cannot happen (i.e. safety properties) rather than what must happen
- ◇ Liveness requirements often manifest themselves as making some sort of *fair choice* between internal state transitions.
- ◇ In nondeterministic operational models the *eventuality* of something good happening depends on the nondeterminism being resolved *fairly*.
- ◇ Choice of temporal logic as a suitable means for expressing and reasoning about such fairness properties.

Fair objects: Fairness from TLA

- **Weak fairness** states that if an action is continually enabled then it will be eventually carried out.
- **Strong fairness** guarantees the eventual execution of an action when that action is enabled an infinite number of times (though not necessarily continuously).
- **Possible Fairness** deals with the case when it is always possible for an action to be enabled (by following a certain sequence of internal actions) yet the action cannot be guaranteed to be executed through the use of strong fairness.
- Unfortunately, TLA does not provide the means for easily constructing and validating initial customer requirements
- By combining TLA and operational (object-oriented) semantics we can alleviate these problems.

Fair objects

- ◇ Combine temporal semantics and object-oriented concepts in a complementary fashion.
- ◇ The underlying semantic model is that of an object as a labelled state transition system (O-LSTS).
- ◇ three temporal modalities which provide useful high-level specification mechanisms for requirements modelling: *fair servers*, *progression* and *politeness*.
- ◇ A *fair object* is one which cannot introduce deadlock (or livelock) into a system through continually refusing to execute a method request.
- ◇ A *fair object* has the property that all its external services are always eventually enabled.
- ◇ An object must be viewed as an open system which relies on its environment to ensure that its liveness properties are satisfied.
- ◇ When these objects are composed, the resulting composition must be checked to ensure that the liveness properties specified for each object are preserved: these objects are *polite*.


POTS
regard=on
listen=ringing
dialin
otherDown
regard=on
listen=silent
lift
drop
regard=off
listen=ready
regard=on
listen=ready
lift
lift
drop
dial(ID)
listen:signal
regard:on-off

phone (POTS) Fair Object

- The operational requirements : There are eight states in the system and the initial state, identified by an incoming arrow which is not rooted from another state, is `on-silent`.

- Safety requirements: Safety requirements are specified as state invariant properties.

  The telephone is specified as consisting of two object components: `regard` and `listen`. A state invariant which defines a relation between these components will include the following requirement:

  > *Always*, if I am `talk`ing with someone then the telephone must be `off` hook
  >
  > `□ ((listen = talk) => (regard = off))`

- Fairness requirements:

  > I will *eventually* leave the state `off-connecting` even if I do not force the state transition myself.
  >
  > $WF(noconnection)$.

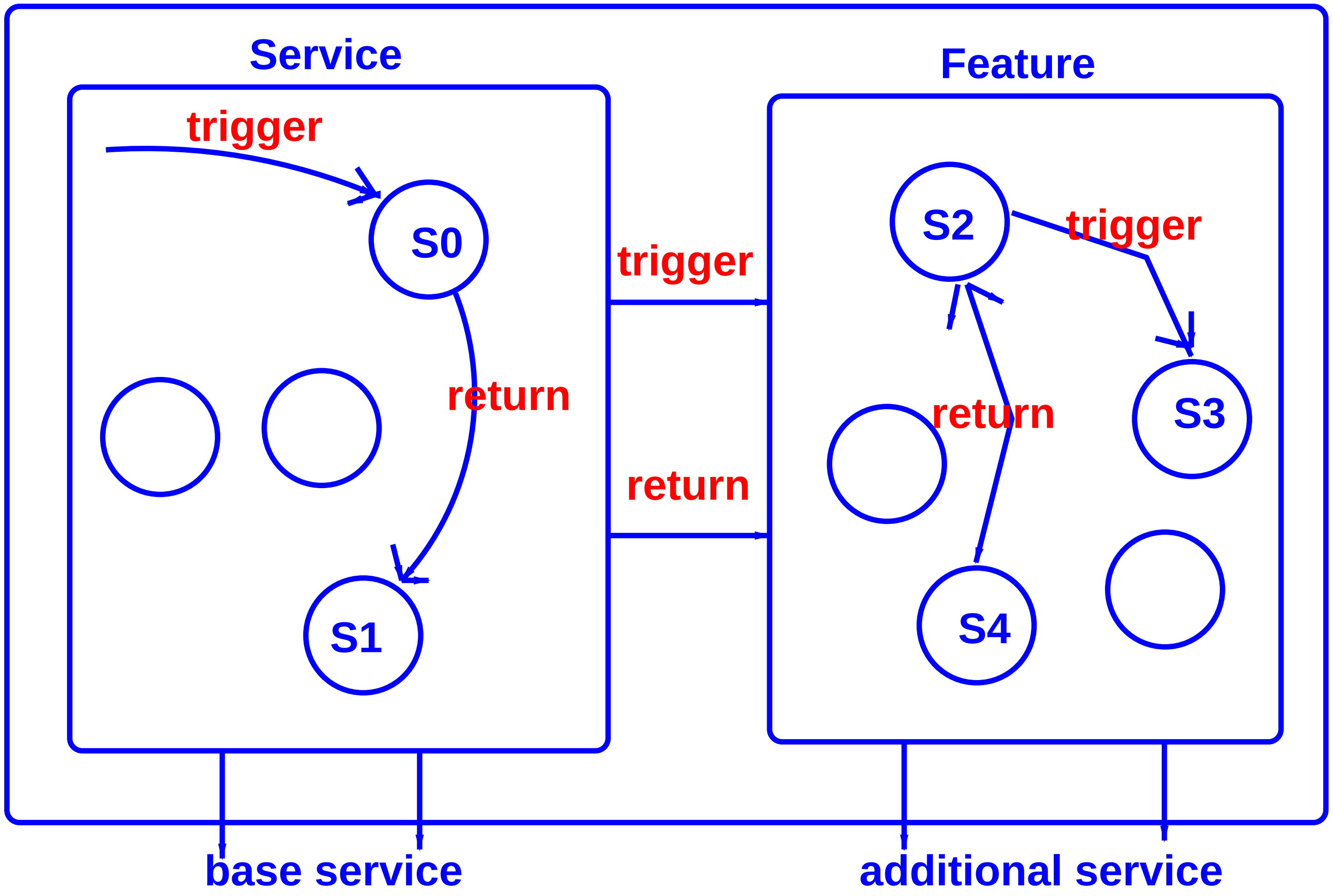
The need for client-server eventuality classes

- ◇ **Immediately Obliged —** A client may require that a service request be serviced immediately.
- ◇ **Eventually Obliged —** A client may require that a service is carried out eventually.
- ◇ **Immediately Conditional —** A client may wish a service to be performed immediately but if it cannot be done without delay then it must be informed so that it can attempt to do something else.
- ◇ **Eventually Conditional —** The service is required eventually but if it cannot be guaranteed in a finite period of time then the client must be informed so that something else can be done.
- ◇ **Unconditional —** The client wants the service but places no eventuality requirements on when (or if) the service is performed.

Using liveness to classify fair object interactions

- **Independent** - if they do not synchronise on any actions
- **Perfect friends** - if they do synchronise on actions, but the internal liveness constraints do not need to be strengthened to meet the local liveness requirements of the composed object
- **Friends** - if they synchronise on actions, but the internal liveness constraints of actions in one or both objects need to be changed to meet the local liveness requirements of the composed object
- **Politicians** - if the local liveness requirements of the composed object cannot be met by changing the internal liveness constraints in either object unless some additional resolution mechanism is used
- **Enemies** - if the local liveness requirements of the composed object cannot be met by changing the internal liveness constraints in either object or by using any additional resolution mechanism

The base trigger class

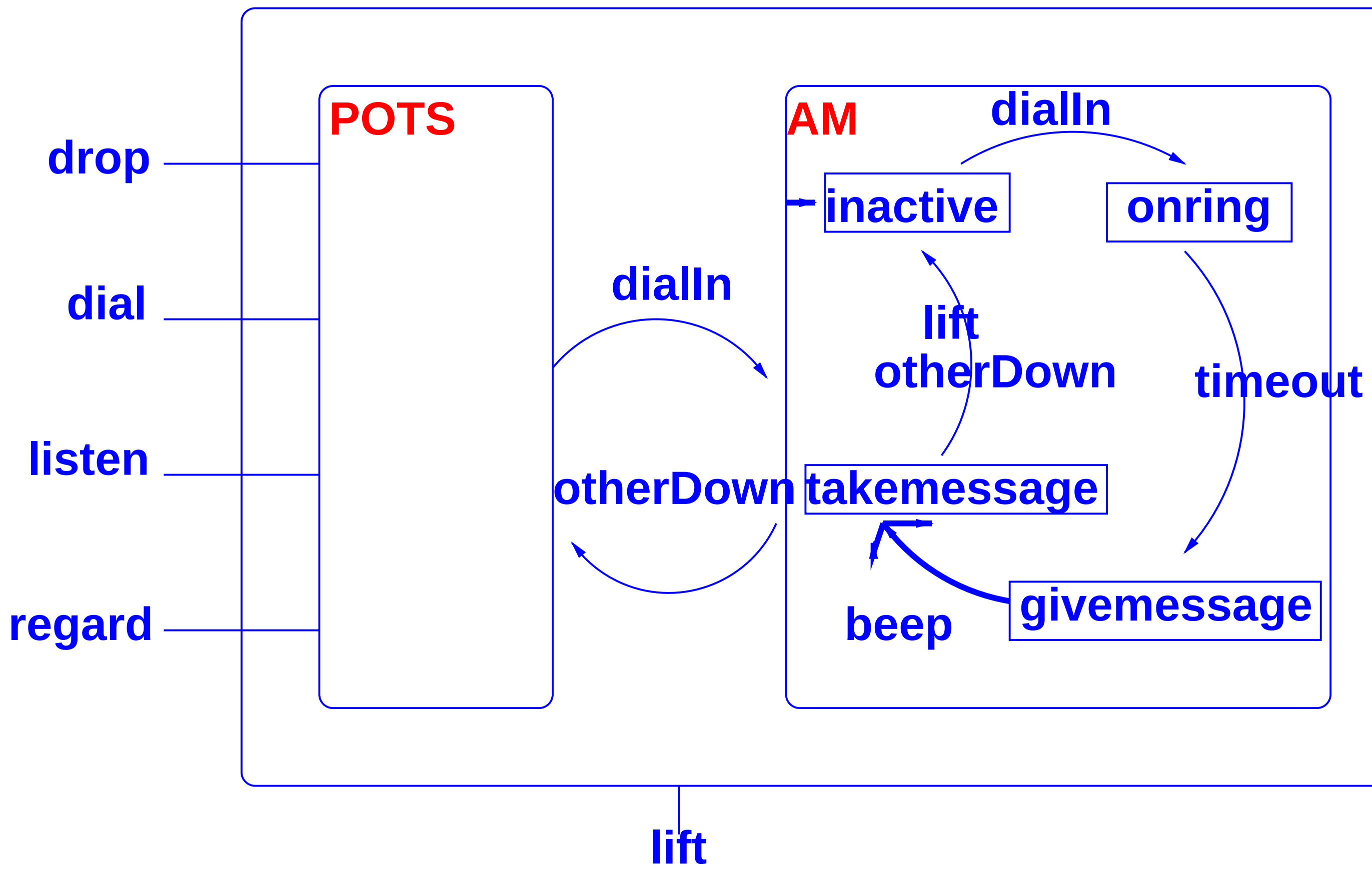


What is triggering?

- State S0 is the state of the service in which the feature has just been **trigger**ed (and is henceforth known as an **trigger state**).
- State S1 is the state of the service to which the feature **return**s control to the service (and is henceforth known as a **return state**).
- State S2 is the initial (inactive) state of the feature.
- State S3 is the state in which the feature is initially active after triggering (and is henceforth known as an **entry state**).
- State S4 is the final active state in which the feature can return control to the service (and is henceforth known as an **exit state** of the feature.)
- There may only be one of each of the states S0 to S4, although they do not necessarily have to be distinct. So, for example, S0 and S1 may be the same state.
- There may only be one trigger action and one return action, although they do not necessarily have to be distinct.
- The service and feature must synchronise on the trigger and the return actions.
- The feature may add additional services (as actions at its external interface) but these must not be found in the base services offered by the service.

An answer machine as a triggered fair object

- Answermachine is triggered by dialIn : Onringing can occur only after a dialIn.
- OnRinging state is refined into three states.
- A reasonable *liveness* requirement is that when I ring someone with an answering machine I will eventually `talk` with them or get to leave a message.
- This requirement is quite naturally specified by making the answering machine a fair object, which guarantees the eventual execution of a `timeout` and `beep` provided no external actions return control to the `Phone`.


POTS
AM
drop
dial
listen
regard
lift
dialIn
otherDown
dialIn
inactive
onring
lift
otherDown
timeout
takemessage
givemessage
beep

Classification of triggering concept

- Variation 1: multiple trigger and return instances
- Variation 2: multiple return state
- Variation 3: different triggers and returns
- Variation 4: no loss of control

Classification of DialIn triggered features

- ◇ TWC, CH — variation 3 (single trigger, multiple trigger instances, multiple returns)
- ◇ CID, CLOG — variation 4
- ◇ AM — variation 2 (single trigger, multiple return states, multiple returns)
- ◇ ICS — this does not correspond to any of our variations. The problem with this feature is that it appears to filter out some of the allowable traces in the original service behaviour and so the new service+feature model is definitely not a refinement of the old service model.

Classifying triggered interactions

- ◇ Two variation4 features cannot interact when they share the same trigger action provided they cannot block the trigger action from being executed and they have different return actions.

- ◇ Two variation3 features cannot interact on different trigger events provided they are guaranteed to eventually return control to the original service.

- ◇ If F1,F2 and F3 are triggered features then they cannot interact when all three are implemented in the same system provided no two of them interact in a *pair-wise* fashion

Conclusion

- ◇ Ensuring that the liveness properties of objects are preserved under composition is to allow only a weak form of object composition in which the liveness properties of the objects are guaranteed to be preserved.
- ◇ The liveness properties of the com posed object in this case are easy to represent i/ n TLA as the logical conjunction of the liveness properties of each object.
- ◇ In general, however, we require stronger forms of object composition in which there is communication (through synchronisation) between the composed objects.
- ◇ Ongoing work on proofs and categorisation of features