Transition-based Qualitative Simulation

John M Gooday and Anthony G Cohn
Artificial Intelligence Division
School of Computer Studies
University of Leeds
Leeds LS2 9JT, UK
{gooday,agc}@scs.leeds.ac.uk

Abstract

In this paper we present an event-based approach to qualitative simulation. We suggest that the behaviour of a system with time is best measured in terms of the landmark events that occur i.e. events that result in interesting changes to the system being modelled. For us, a behaviour model corresponds not to a sequence of qualitative state descriptions but to a set of event sequences — the things that actually happen to the system rather than the way it happens to be at certain times.

Although we have a simple implementation of our system, our primary purpose in developing it is to derive a high level, event-based, nonmonotonic language for specifying qualitative simulation systems. We not only illustrate how a qualitative simulation program can be directly specified (and implemented) in our language, we also sketch how qualitative simulation systems from the literature can be defined and reconstructed in our calculus.

Introduction

Qualitative simulation is a well-established artificial intelligence technique for modelling and predicting the behaviour of physical systems. Programs such as QSIM (Kuipers 1994) derive behaviour models from an initial qualitative description of a system and a set of constraints that specify how individual parameter values within the system might change. By varying system parameters in accordance with the constraints, a sequence of time-ordered snapshots (or qualitative states) can be generated. The set of all possible such sequences (or *histories*) forms the complete *behaviour tree* of the system being modelled. Other Qualitative Reasoning systems (Forbus 1990; De Kleer & Seely Brown 1984), alternatively, generate an envisionment giving all possible legal transitions between states.

In this paper we show how an alternative approach to qualitative simulation can be developed by using an event-based approach. We suggest that the behaviour of a system with time is best measured in terms of the landmark events that occur i.e. events that result in interesting changes to the system being modelled. For us, a behaviour model corresponds not to a sequence of qualitative state descriptions but to a set of event sequences — the things that actually happen to the system rather than the way it happens to be at certain times. Of course, this is closely related to the more traditional view mentioned above, and indeed they might be considered duals: in our representation events are explicit and states implicit, while the reverse is true in most other Qualitative Reasoning systems. Whereas QSIM presents the user with a sequence of qualitative states from which an event model needs to be extracted, our approach provides an event model from which a corresponding sequence of qualitative states can be easily derived should they be required.

Although we have a simple implementation of our system, our primary purpose in developing it is to derive a high level, event-based, non monotonic lan-

guage for specifying qualitative simulation systems. In this paper we not only illustrate how a qualitative simulation program can be directly specified (and implemented) in our language, we also sketch how qualitative simulation systems from the literature can be defined and reconstructed in our calculus. The language was originally developed as a calculus for reasoning about the classic challenges in the nonmonotonic reasoning literature (Gooday 1994; Gooday & Galton 1996). Although the nonmonotonicity of the calculus is only exploited rather indirectly in this paper, potentially it provides a general mechanism for reasoning about persistence and ramification in a qualitative simulator.

The rest of the paper is organized as follows. First the basic elements of *Transition Calculus* are introduced. Then we show how this can be used as the basis of a qualitative simulator. In order to provide increased expresivity, a number of extensions to the original calculus are introduced and explained. The use of Transition Calculus for simulation is illustrated by constructing a *transition network* for a simple bathtub example. In the final parts of the paper we sketch the reconstruction of two earlier qualitative simulation systems and conclude with suggestions for further work.

Transition Calculus

Transition Calculus is a formalism for reasoning about action and change and is based on the idea that interesting events can be characterised solely in terms of the state changes associated with them. For example, the event of opening a valve could be characterised by a change of state in the valve i.e. from being closed immediately before the event to being open immediately afterwards.

In order to describe the world, Transition Calculus employs a simple *state language*. A *state* in Transition Calculus is either an atom that serves to represent some object in the world or property thereof, or a negated atom. For example, the following are all states: $Valve(InputValve_1)$, $Open(InputValve_1)$, $\neg Operating(Machine_7)$. The first two are atoms, the third is a negated atom. A *stateset* is simply a set of states.

Often, if a particular state holds then certain other states will be excluded from holding at the same time. For example the states $Open(InputValve_1)$ and $\neg Open(InputValve_1)$ could not be used together to describe any consistent world. More subtly, $Open(InputValve_1)$ and $Closed(InputValve_1)$ are also mutually exclusive. Transition Calculus allows constraints to be placed on which states can hold simultaneously via the notion of *compatibility*. Only state sets that are compatible are legal in Transition Calculus. The following four conditions are used to determine whether a state set is or is not compatible:

1. $\{s\}$ is compatible
2. $\{s, \overline{s}\}$ is not compatible
3. if $S = \{s_1, ..., s_n\}$ is compatible then every subset of S must be compatible
4. if $S \cup \{\overline{s}\}$ is not compatible and $S \cup \{s, \overline{s'}\}$ is not compatible then $S \cup \{\overline{s'}\}$ is not compatible

where $\overline{s}$ denotes the complement of s:

$$\overline{s} = \neg s$$
$$\overline{\neg s} = s$$

Obviously, an empty state set or one consisting of a single state must be compatible, just as a state set that contains two states, one of which is the negated form of the other, must be incompatible. A state set can only be compatible if it contains no incompatible subsets, hence the third condition. (Conversely, it is not necessary for every subset of an incompatible state set to be incompatible.) The final condition simply represents a kind of cut rule (Gentzen 1955). In addition to these four conditions we also allow particular state sets to be flagged as incompatible. For example,

$$Incompatible(\{Open(Valve_1), Closed(Valve_1)\})$$

This allows us to to recognise incompatibilities that would otherwise be impossible to represent.

So far we have looked only at static aspects of the world. We now turn our attention to events. An event type is represented using a *transition schema*, which we write as

$$\langle\langle S_1, S_2 \rangle\rangle$$

where S_1 is the state set that holds immediately before the transition (the *precondition*); and S_2 is the state set that holds immediately after the transition (the *postcondition*). Note that both S_1 and S_2 must be compatible state sets in order for the transition schema to be legal.

A transition schema $\langle\langle S_1, S_2 \rangle\rangle$ is said to be a subtype of the schema $\langle\langle S_1', S_2' \rangle\rangle$, written

$$\langle\langle S_1, S_2 \rangle\rangle \sqsubseteq \langle\langle S_1', S_2' \rangle\rangle,$$

if $S_2' \subseteq S_2$ and $S_1' \subseteq S_1$. Consequently, every schema is a subtype of $\langle\langle \{\}, \{\} \rangle\rangle$.

Two or more transition schemas $T_1 ... T_n$ can be sequentially composed to form a *transition sequence* in which the first transition occurs immediately prior to the second, written $T_1 \circ ... \circ T_n$, providing that the following composition rule is obeyed.

(Composition Rule) A transition sequence $\langle\langle S_{1,1}, S_{2,1}\rangle\rangle \circ ... \circ \langle\langle S_{1,n}, S_{2,n}\rangle\rangle$ may only be formed if $S_{2,i} \cup S_{1,i+1}$ is compatible for every i in the range $1 \leq i < n-1$.

Qualitative simulation with Transition Calculus

It is relatively easily to model a qualitative system using Transition Calculus. The following simple example[1] gives an idea of how one might go about describing such systems. Figure 1 shows a tank which can be filled from a pipe controlled by valve 1. Valve 2 controls the emptying of the tank. The level in the tank changes between the values empty, midway and full depending on the value of the net flow of liquid into the tank. If valve 1 is open and valve 2 is closed then the net flow is positive and causes the level to rise. If valve 1 is closed and valve 2 is open then the net flow is negative and the level falls. No change in the level occurs if the net flow is zero (when both valves are closed). Finally, if both valves are open the net flow of liquid cannot be determined and we are unable to infer how the level will change. Three kinds of changes can

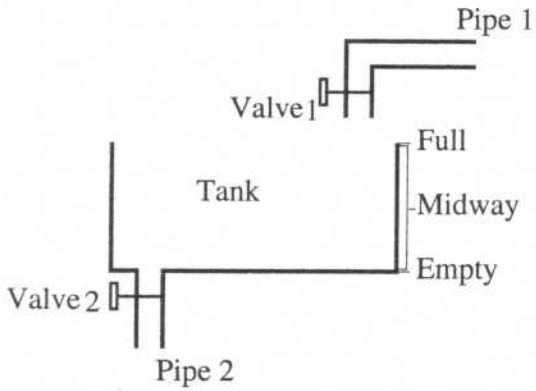

Figure 1: Simple physical system

occur in the system: changes to the state of the valves (open or closed); changes to the net flow of liquid in to the tank; and changes to the level of liquid. All of the events that could potentially cause these changes can be modelled directly using transition schemas.

We begin with the events representing changes to the liquid level. These can be written as follows[2]:

$$\langle\langle \{NetFlow(x, Plus), Level(x, Empty)\}, \{Level(x, Midway)\}\rangle\rangle \quad (1)$$

$$\langle\langle \{NetFlow(x, Plus), Level(x, Midway)\}, \{Level(x, Full)\}\rangle\rangle \quad (2)$$

$$\langle\langle \{NetFlow(x, Minus), Level(x, Full)\}, \{Level(x, Midway)\}\rangle\rangle \quad (3)$$

$$\langle\langle \{NetFlow(x, Minus), Level(x, Midway)\}, \{Level(x, Empty)\}\rangle\rangle \quad (4)$$

We have chosen to use the state $NetFlow(x, y)$ to represent the value of the net flow (y) of liquid into x, and $Level(x, y)$ to represent the liquid level (y) in x.

The next set of transition schemas models the possible changes in net flow and valve state:

$$\langle\langle \{NetFlow(x, Plus), Valve(y, z, x), Open(y)\}, \{\neg Open(y), NetFlow(x, 0)\}\rangle\rangle \quad (5)$$

$$\langle\langle \{NetFlow(x, Minus), Valve(y, x, z), Open(y)\}, \{\neg Open(y), NetFlow(x, 0)\}\rangle\rangle \quad (6)$$

$$\langle\langle \{NetFlow(x, Plus), Valve(y, x, z)\}, \{Open(y), NetFlow(x, Unknown)\}\rangle\rangle \quad (7)$$

$$\langle\langle \{NetFlow(x, Minus), Valve(y, z, x)\}, \{Open(y), NetFlow(x, Unknown)\}\rangle\rangle \quad (8)$$

$$\langle\langle \{NetFlow(x, 0), Valve(y, x, z), \neg Open(y)\}, \{Open(y), NetFlow(x, Minus)\}\rangle\rangle \quad (9)$$

$$\langle\langle \{NetFlow(x, 0), Valve(y, z, x), \neg Open(y)\}, \{Open(y), NetFlow(x, Plus)\}\rangle\rangle \quad (10)$$

Here, we have used the state $Valve(x, y, z)$ to represent the information that valve x takes input from source y and feeds an output z. $Open(x)$ is intended to denote that x is open.

Consider the state $Level(x, y)$ in the above schemas. We can think of $Full$, $Midway$ and $Empty$ as defining a qualitative parameter space for y. It is necessary to ensure that for any particular x, y takes one of these parameter values only i.e. we need to ensure that state sets containing pairs such as $\{Level(Tank, Full), Level(Tank, Empty)\}$ are outlawed. This is easily accomplished with the following compatibility conditions.

$$Incompatible(\{Level(x, Empty), Level(x, Midway)\})$$
$$Incompatible(\{Level(x, Empty), Level(x, Full)\})$$
$$Incompatible(\{Level(x, Full), Level(x, Midway)\})$$

Similarly, for states of the form $NetFlow(x, y)$ it is necessary to ensure that for any x, y takes only one value from its parameter space of $Plus, Minus, 0, Unknown$:

$$Incompatible(\{NetFlow(x, Plus), NetFlow(x, Minus)\})$$
$$Incompatible(\{NetFlow(x, Plus), NetFlow(x, 0)\})$$
$$Incompatible(\{NetFlow(x, Plus), NetFlow(x, Unknown)\})$$
$$Incompatible(\{NetFlow(x, Minus), NetFlow(x, 0)\})$$

[1] This example is provided primarily to explicate the Transition Calculus rather than as a serious QR model – various simplifying assumptions have been made in the modelling process.

[2] Note that we use lower case letters to denote variables. Variables may be thought of as universally quantified and their scope as being the entire transition schema.

$Incompatible(\{NetFlow(x, Minus),$
$NetFlow(x, Unknown)\})$
$Incompatible(\{NetFlow(x, 0),$
$NetFlow(x, Unknown)\})$

Finally, in this model, we place three further constraints on the net flow value. The net flow into a tank cannot be 0 if any valve connected to it is open:

$Incompatible(\{NetFlow(x, 0),$
$Valve(y, z, x), Open(y)\})$
$Incompatible(\{NetFlow(x, 0),$
$Valve(y, x, z), Open(y)\})$

The net flow into a tank cannot be positive if it has an open output valve:

$Incompatible(\{NetFlow(x, Plus),$
$Valve(y, z, x), Open(y)\})$

The net flow into a tank cannot be negative if it has an open input valve:

$Incompatible(\{NetFlow(x, Minus),$
$Valve(y, x, z), Open(y)\})$

The transition schemas above describe single events in the system and can be viewed as isolated behaviour fragments. Our task now is to link these fragments together in order to model the behaviour of the system as a whole. We do this by constructing a *transition network* — a directed graph in which the nodes represent individual transition schemas and the arcs indicate which schemas may be sequentially composed and in what order. A path through such a network corresponds to a potential behaviour of the system. In order to create a transition network we take each pair of transition schemas in turn and determine whether or not they can form a legal transition sequence. For each legal sequence $T_1 \circ T_2$ we draw a directed arc from T_1 to T_2. The resulting structure is the full transition network for the system.

It should be noted that the transition schemas that we have used to describe a system are, in some cases, incomplete. This is useful in that it allows us to write quite general schemas and also prevents us from having to repeat in the postcondition states present in the precondition that are unchanged (e.g. $Level(x, y)$ in schemas (1), (2), (3) and (4) above). This presents a problem when constructing the transition network. Consider a sequence made up from (1) and (4):

$\langle\langle\{NetFlow(x, Plus), Level(x, Empty)\},$
$\{Level(x, Midway)\}\rangle\rangle \circ$
$\langle\langle\{NetFlow(x, Minus), Level(x, Midway)\},$
$\{Level(x, Empty)\}\rangle\rangle$

This forms a legal transition sequence as

$\{Level(x, Midway)\} \cup$
$\{NetFlow(x, Minus), Level(x, Midway)\}$

is compatible. However, this sequence suggests that the net flow into x somehow changes from $Plus$ to $Minus$ during first transition. This mysterious change is clearly not a direct result of the event modelled by (1) and we cannot explain it in terms of known events. In order to eliminate arcs in the transition network resulting from such spontaneous state changes we require a method for giving priority to the most plausible transition sequences — those in which no unexplained state changes occur. A virtually identical problem can be found in the literature on nonmonotonic reasoning about action and change. Given an initially incomplete scenario involving action and change it is usually possible to produce a number of alternative (completed) models in which different state changes occur. The difficulty is to select the most plausible model. In (Gooday & Galton 1996) we showed how this problem could be overcome using a specially developed model preference criteria that minimizes unexplained state change. We can use this approach to eliminate arcs in the transition network that correspond to unexplained state changes.

Eliminating unexplained state change

In order to overcome the above problem, we must first automatically complete the transition sequences and then evaluate the completed sequences against a set of preference criteria in order to determine whether they are plausible or not. Only pairs of nodes in the transition network representing schemas that can form plausible sequences will have arcs linking them.

A transition sequence is said to be *complete* iff

for each $s \in \bigcup_{i=1}^{n} (S_{1,i} \cup S_{2,i})$
for all $j, 1 \leq j \leq n$, and for $k = 1, 2$
either $s \in S_{k,j}$
or $\overline{s} \in S_{k,j}$

We say that a transition sequence

$$T = \langle\langle S_{1,1}, S_{2,1}\rangle\rangle \circ ... \circ \langle\langle S_{1,n}, S_{2,n}\rangle\rangle$$

is a model for the (possibly incomplete) transition sequence $T' = \langle\langle S'_{1,1}, S'_{2,1}\rangle\rangle \circ ... \circ \langle\langle S'_{1,n}, S'_{2,n}\rangle\rangle$ if and only if

1. T is a completion of T' (generated from T' by arbitrarily adding either s or $-s$ for every $s \in S'_{1,1} \cup S'_{2,1} \cup ... \cup S'_{1,n} \cup S'_{2,n}$ to any $S_{i,j}; i = 1, 2; j = 1...n$ containing neither).
2. T obeys the composition rule: for all $1 \leq i \leq n-1, S_{2,i} \cup S_{1,i+1}$ must be compatible.

We define a penalty assignment function, η, that returns the *penalty* associated with a model. This is an integer value calculated according to the following scheme:

For each $s \in \bigcup_{i=1}^{n} (S'_{1,i} \cup S'_{2,i})$,
for each $k = 1, ..., n$,

(Rule 1) If $s \in S_{1,k}$ and $s \in S_{2,k}$ then no penalty.

(Rule 2) If $s \in S'_{1,k}$ and $\overline{s} \in S'_{2,k}$then no penalty.

(Rule 3) If $s \in S_{1,k}$ and $\overline{s}\} \cup S'_{1,k}$ is incompatible then no penalty.

(Rule 4) A penalty of 1 is awarded for each instance of every other case.

The penalty scheme checks each state in every state-set of the model and attempts to explain its presence according to three basic principles: inertia, motivated action and ramification. For every state that cannot be explained, a penalty point is awarded to the model. A plausible transition sequence will yield at least one model with no penalty points i.e. at least one model that adheres to all three principles. Rules (1) to (3) above check for the three different kinds of explanation:

(Rule 1) Inertia. If a state appears unchanged in both preconditions and postconditions of a schema then this satisfies the general principle of inertia: things tend to remain the same if they can.

(Rule 2) Motivated action. If a state appears in the precondition of the schema and its complement appears in the postcondition then this change should have been explicitly motivated: i.e it should be forced by the original, uncompleted, sequence.

(Rule 3) Ramification. Quite often a state change that may be explained by an action or observation will, in turn, force additional state changes via compatibility conditions. Rule (3) checks for this.

Additional constraints

Compositional constraints

In order to increase expresivity further, we introduce an additional kind of constraint: *compositional restrictions*. These allow us to explicitly state that transitions of a certain specified type may only be followed by transition of some other specified type. Such constraints are not part of Transition Calculus itself, they are meta-level restrictions that can be applied to the completed models of transition sequences.

We write compositional restrictions in the following form:

$$CCon(T : \{T_1, ..., T_n\})$$

where $T, T_1, ..., T_n$ are transition schemas and $n \geq 1$ We say that a model $T_1 \circ ... \circ T_n$ for some transition sequence $T'_1 \circ ... \circ T'_n$ satisfies compositionality restrictions for every pair T_i, T_{i+1}; $i = 1, n-1$ for which there exists a compositional restriction $CCon(T_1^*, TS)$ such that $T_1^* \sqsubseteq T_1$ there is some element T_2^* of TS such that $T_2^* \sqsubseteq T_2$.

For example, in our tank system we might reasonably want to specify that transitions in which the net flow into a tank changed from 0 to $Plus$ should be immediately followed by transitions in which the level of the tank increases or remains full. We can do this by adding the following compositional restrictions:

$CCon(\langle\langle\{NetFlow(x,0)\}, \{NetFlow(x, Plus)\}\rangle\rangle :$
$\{\langle\langle\{Level(x, Empty)\}, \{Level(x, Midway)\}\rangle\rangle,$
$\langle\langle\{Level(x, Midway)\}, \{Level(x, Full)\}\rangle\rangle,$
$\langle\langle\{Level(x, Full)\}, \{\}\rangle\rangle\})$

Deriving more specific event schemas

The schemas that we have used to model our example system are quite general and make use of states that contain variables. This means that we can use these same schemas to describe transitions in more complicated systems than our simple example (e.g. a system in which there are many tanks and valves interconnected)[3]. However, when considering a particular system it is useful to be able to derive from these general schemas more specific schemas in which the relevant variables have been assigned the names of objects and values that actually feature in the system. These instantiated schemas provide more specific information and it is therefore desirable to base the transition network on these. By associating a particular type of state with the range of values and or objects this is easy to accomplish. In our example system there are only two valves so all states of the type $Valve(x, y, z)$ can take only two value sets corresponding to these. Either (x, y, z) takes the value set $(V_1, Pipe_1, Tank)$ or $(V_2, Tank, Pipe_2)$. We introduce state constraints to handle this. These are written

$SCon(S(x_1, ..., x_n) :$
$\{\langle v_{1,1}, ..., v_{1,n}\rangle, ..., \langle v_{1,m}, ..., v_{1,m}\rangle\})$

which force the variables in the state $S(x_1, ..., x_n)$ to take one of the value sets given.

Constructing the network

In order to create a preferred transition network for a system described by a set of transition schemas $Trans$, a set of compatibility constraints $Const$, a set of state

[3] In fact, in order to fully generalise our tank example we must take into account multi-valve systems in which opening or closing a single valve may not change the net flow in an attached tank. We do this by adding two further transition schemas:

$\langle\langle\{Valve(x, y, z), Open(x)\}, \{\neg Open(x)\}\rangle\rangle$ (11)
$\langle\langle\{Valve(x, y, z), \neg Open(x)\}, \{Open(x)\}\rangle\rangle$ (12)

constraints *Stat* and a set of compositional restrictions *Comp* we use the following procedure.

1. Generate a set of system-specific schemas, $Trans'$ from *Trans* according to the constraints in *Stat*.
2. Create a node for each schema in $Trans'$ and label it with the schema.
3. For each ordered pair $\langle T_1, T_2 \rangle$; $T_1 \in Trans'$, $T_2 \in Trans'$: such that a plausible model M of $T_1 \circ T_2$ can be generated and M satisfies the compatibility constraints in *Const* and the compositionality restrictions in *Comp*, draw a directed arc from the node labelled by T_1 to the node labelled by T_2.

Applying this procedure to our tank example produces the transition network shown in figure 2 (in fact, this network was generated using a Prolog implementation of the above procedure (and drawn by hand)).

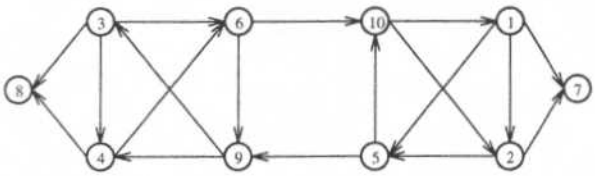

Figure 2: Transition network for the watertank example

Given the Transition Network, it is possible to derive a behaviour tree for the system. Each non-cyclic path through the graph that has a corresponding plausible model can be viewed as a branch of the behaviour tree. By extracting all such maximal paths from the graph, we obtain the tree.

Rational reconstructions

In this section we show how some Qualitative Reasoning systems from the literature can be reconstructed in the Transition Calculus. The first system we will consider is a qualitative spatial simulator developed here at Leeds. In the second subsection below we will sketch how to reconstruct the well known QSIM system (Kuipers 1994).

Reconstructing QSSIM: A Qualitative spatial simulator

In (Cui, Cohn, & Randell 1992a; 1992b) a qualitative spatial simulator was presented which was based upon the RCC spatial calculus presented in (Randell, Cui, & Cohn 1992). In this system, regions are the primitive spatial entities and various jointly exhaustive and pairwise disjoint (JEPD) sets of relations which may hold between pairs of regions are defined in terms of a very small number of primitives. One of the simplest of these, which is defined entirely in terms of a primitive notion of connection, C(x,y), which holds when the two regions x and y are connected (there is not space to discuss the precise semantics here) is a system of 8 relations, known as RCC8[4]; figure 3 presents 2D ex-

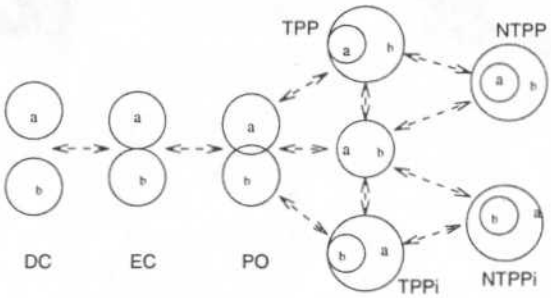

Figure 3: The continuous transitions (conceptual neighbourhood) for RCC8

ample configurations for these 8 relations (though the calculus is not limited to 2D). The figure also presents the continuity network (sometimes known as a conceptual neighbourhood) which shows which transitions between relations are possible assuming continuous motion and/or deformation of the regions. The spatial simulator is based around these networks: each pair of regions defines a kind of quantity space whose structure is given by this network, with the relation names being the values in the space. The essence of the simulator is to take a state set defined as a set of ground atoms whose relation names are RCC relations and to build a set of next state sets by forming the cross product of all possible transitions given by considering the adjacent values in each quantity space. Domain specific modelling is achieved by *intrastate constraints* which specify conditions which must always hold in any state set, and *interstate constraints* which effectively rule out certain transitions in a particular quantity space, depending on particular conditions appertaining in the originating state set. Add and delete rules are also provided for creating new regions with specified relations to existing regions under certain conditions, and similarly deleting regions under specified conditions. The paper shows how the phagoctyosis of a unicellular organism can be modelled with this system.

[4]There are simpler RCC systems (e.g. RCC 5 which corresponds to mereology or simple set theoretic relations) and many more expressive RCC systems with much larger sets of JEPD relation sets, e.g. allowing non convex regions to be reasoned with (Gotts 1994; Cohn, Randell, & Cui 1995). However, RCC8 will suffice to explain qualitative simulation in RCC.

It is not difficult to see how this system can be reconstructed within the Transition Calculus. The quantity space/conceptual neighbourhood diagram is modelled by a set of local transition schemas. Since there are 11 links in the RCC8 conceptual neighbourhood, there are 22 such transition schemas, e.g. the ones involving transitions from EC are

$\langle\langle\{Holds(\mathsf{EC}, x, y)\}, \{Holds(\mathsf{DC}, x, y)\}\rangle\rangle$

and

$\langle\langle\{Holds(\mathsf{EC}, x, y)\}, \{Holds(\mathsf{PO}, x, y)\}\rangle\rangle$[5].

Intrastate constraints are easily modelled by incompatibility statements; e.g. the intrastate constraint $\mathsf{NTPP}(E, A)$ is modelled by the statment $Incompatible(\text{–}Holds(\mathsf{NTPP}, E, A))$.

Interstate constraints are slightly trickier; however the composition restrictions described earlier in this paper can be used for this purpose. E.g. consider the QSSIM interstate constraint: $\Phi \rightarrow (\mathsf{EC}(F, A) \Rightarrow \mathsf{PO}(F, A))$ which ensures that as soon as Φ and $\mathsf{EC}(F, A)$ hold, then in the following state set, $\mathsf{EC}(F, A)$ will transition to $\mathsf{PO}(F, A)$; this can be modelled as a composition restriction thus:

$CCon(\langle\langle\{\}\{\Phi, Holds(\mathsf{EC}, F, A)\}\rangle\rangle :$
$\langle\langle\{Holds(\mathsf{EC}, F, A)\}\{Holds(\mathsf{PO}, F, A)\}\rangle\rangle)$

This ensures that any event which results in both Φ and $\mathsf{EC}(F, A)$ being true, will be immediately followed by another event in which $\mathsf{EC}(F, A)$ transitions to $\mathsf{PO}(F, A)$, as required.

Add and delete rules are logically rather intricate, since they involve changing the universe of discourse: new individuals can be created and existing ones can cease to exist! Since the completion mechanism of the Transition Calculus completes across all state sets with respect to a fixed set of propositions, this creates problems[6].

There are a number of possible approaches to modelling this kind of situation formally; the approach we will take here is as follows. Each region r which exists in a state set must have a proposition $Exists(\mathrm{r})$ asserted in that state set. An add rule is modelled as a transition schema. E.g. consider the add rule:

add V *with* relations $\mathsf{TPP}(V, A) \wedge \mathsf{TPP}(F, A)$
when $\mathsf{TPP}(F, V)$

This can be modelled thus:

$\langle\langle\{\text{–}Exists(V), Holds(\mathsf{TPP}, F, V)\},$
$\{Exists(V), Holds(\mathsf{TPP}, V, A), Holds(\mathsf{TPP}, F, A)\}\rangle\rangle$

Forward and backward completion will propagate the relevant statements about the existence of V to past and future state sets appropriately.

A delete rule such as *delete* E *when* $\mathsf{TPP}(E, F)$ is modelled thus:

$\langle\langle\{Exists(E), Holds(\mathsf{TPP}, E, F)\}, \{\text{–}Exists(E)\}\rangle\rangle$

Again, forward and backwards completion will handle the existence of E in past and future state sets appropriately. However the same completion mechanism will also propagate $\mathsf{TPP}(E, F)$ forwards even though E no longer exists! Similarly in the add rule above, $\mathsf{TPP}(V, A)$ will be propagated backwards before V existed! To eliminate such nonsense propositions we adopt the following technique. A special symbol None will be introduced as a 9th possible first argument to the $Holds(,,)$ predicate and we introduce an incompatibility statement

$Incompatible(\{\text{–}Exists(x),$
$Contains(Holds(\alpha, v, w), x)\})$

for each RCC8 relation α, where $Contains(y, x)$ is a special meta predicate which is true when the expression y is true in the state set and contains the term x[7]. We need to express the fact that if x exists then no $Holds(\mathsf{None}, ,)$ statement can be present containing x:

$Incompatible(\{Contains(Holds(\mathsf{None}, u, v), x),$
$Exists(x)\}).$

This then provides a way of representing the modelling language of QSSIM in Transition Calculus. However our representation task is not complete. In particular we need to represent the composition table (sometimes also known as the transitivity table(Allen 1983)) which is needed in order to check for logical consistency of state set descriptions with respect to the semantics of RCC. The composition table for RCC8 is an 8 by 8 table which specifies the possible relations which may hold between x and z given that the relationship between x and y and y and z is known; i.e. each entry encodes a theorem:

$\forall(x, y, z)[[R_i(x, y) \wedge R_j(y, z)] \rightarrow$
$[R_k^1(x, z) \vee \ldots \vee R_k^n(x, z)]]$

However it is straightforward to model this with incompatibility statements: each RCC8 relation R *not* present in the (R_i, R_j) entry (i.e. which is *not* an R_k^m) gives rise to an incompatibility statement:

$\{Incompatible(Holds(R_i, x, y),$
$Holds(R_j, y, z), Holds(R, x, z))\}.$

Similarly the pairwise disjointness of all the RCC8 relations can be modelled by statements of the form $Incompatible(\{Holds(R_i, x, y), Holds(R_j, x, y)\})$, for each pair of differing RCC8 relations R_i, R_j. The mutual exhaustion of the eight RCC8 relations is also

[5] It turns out that it is easier to represent spatial facts using the notation $Holds(\mathrm{EC,x,y})$ rather than $\mathrm{EC(x,y)}$.

[6] This problem is not unique to the transition calculus; e.g almost every existing QR system assumes the set of modelled items does not change within a model (QSIM's landmark generation is a notable exception).

[7] Logically such an incompatibility statement could always be replaced by a finite number of incompatibility statements which did not make use of *Contains*, but this notation is certainly much more convenient and efficient.

specifiable by a set of incompatibility statements of the form:

$Incompatible(\{Holds(R_1, x, y), ...,$
$Holds(R_7, x, y), \neg Holds(R_8, x, y)\})$

where $R_1 ... R_8$ are the eight RCC8 relations; since each RCC8 relation could take the place of R_8 in the negated $Holds(R_8, x, y)$ literal, there are eight such incompatibility statements. Finally, we need to represent the symmetry and asymmetry of the various RCC relations; for example EC is symmetric and TPP is asymmetric (with inverse TPPi). The appropriate incompatibility statements for these two examples are:

$Incompatible(\{Holds(\mathsf{EC}, x, y), \neg Holds(\mathsf{EC}, y, x)\})$
$Incompatible(\{Holds(\mathsf{TPP}, x, y), \neg Holds(\mathsf{TPPi}, y, x)\})$
$Incompatible(\{Holds(\mathsf{TPPi}, x, y), \neg Holds(\mathsf{TPP}, y, x)\})$

Reconstructing QSIM

The operation of QSIM is well known and will not be described here in detail; we will content ourselves with indicating how the essence of QSIM can be reconstructed in the Transition Calculus. In fact, this is relatively straightforward (at least for QSIM in its simplest form which we will restrict ourselves to here) since Kuipers explicitly describes the operation of the simulator in terms of transitions between values of qualitative quantities.

QSIM distinguishes between states which endure only for an instant and those which endure for longer[8]. We will model this with propositions of the form Instant and ¬Instant. P-transitions (from a point to an interval), e.g.:

P2: $\langle l_j, std\rangle \rightarrow \langle (l_j, l_{j+1}), inc\rangle$

are modelled by transition schemas of the form:

$\langle\langle \{\text{Instant}, \langle l_j, Std\rangle\}, \{\neg\text{Instant}, \langle (l_j, l_{j+1}), Inc\rangle\}\rangle\rangle$

I-transitions (from an interval to a point) are very similar but of course with the polarity of the Instant statements reversed.

QSIM has a variety of constraint primitives which provide the basis for domain modeling; e.g. $ADD(x, y, z)$, M^+(x,y) and $DERIV(x, y)$. Each of these is modelled straightforwardly with a set of incompatibility statements; this follows from theorem 3 of (Kuipers 1994) which states that

> "each QSIM constraint C is associated with a set$\{P_1 ... P_n\}$of provisions that are easily evaluated given a tuple of qualitative values for the variables appearing in C, such that ... $P_1 \vee ... \vee P_n \rightarrow \neg C$".

[8] Although the RCC simulator described above in (Cui, Cohn, & Randell 1992a; 1992b) does not distinguish these two different kinds of temporal state, (Galton 1995) has extended the RCC conceptual neighbourhood to take account of these distinctions which could be exploited if desired.

Each P_i can be represented as an incompatibility statement. Typically, each P_i is a simple equality in the qualitative interval algebra SR1 (Williams 1991). E.g. a condition for $ADD(a, b, c)$ is $[\dot{a}] + [\dot{b}] = [\dot{c}]$. Given the table defining qualitative addition over the quantity space {-,0, + }, this yields the following seven incompatibility statements[9]:

$Incompatible(\{([\dot{a}] = +), ([\dot{b}] = +), \neg([\dot{c}] = +)\})$
$Incompatible(\{([\dot{a}] = +), ([\dot{b}] = 0), \neg([\dot{c}] = +)\})$
$Incompatible(\{([\dot{a}] = -), ([\dot{b}] = -), \neg([\dot{c}] = -)\})$
$Incompatible(\{([\dot{a}] = -), ([\dot{b}] = 0), \neg([\dot{c}] = -)\})$
$Incompatible(\{([\dot{a}] = 0), ([\dot{b}] = 0), \neg([\dot{c}] = 0)\})$
$Incompatible(\{([\dot{a}] = 0), ([\dot{b}] = +), \neg([\dot{c}] = +)\})$
$Incompatible(\{([\dot{a}] = 0), ([\dot{b}] = -), \neg([\dot{c}] = -)\})$

All the other QSIM constraints may be similarly modelled. Two further points are worth noting briefly. Firstly, QSIM allows the possibility of a new qualitative landmark being introduced into a quantity space. This is somewhat akin to the introduction of new regions in the RCC spatial simulator described above and a similar mechanism might be used to handle this situation. Secondly, QSIM allows a model to have different operating regions, each of which has a separate set of modelling constraints. Our approach to this situation would be to build a transition network for each operating region separately; it would be easy to flag automatically (via a proposition in the state set description) which nodes in a network have outgoing arcs to other networks.

Conclusions and future work

In this paper we have described Transition Calculus, a formalism for reasoning about action and change. We extended the expressive power of the calculus and, with the aid of a simple example, showed how it could be used for event-based qualitative simulation. We then explained how our formalism could be used to reconstruct both QSIM and the RCC spatial qualitative simulator.

To the extent that we view Transition Calculus as a tool for reconstructing Qualitative Reasoning systems, it would be interesting to reconstruct other systems. For example, we have done some preliminary work on modelling QPE. The basic mechanism of direct influence resolution can be modelled as the following set of incompatibility statements:

$Incompatible(\{I^+(x), I^-(x), -Ds(x) = ?\})$
$Incompatible(\{I^+(x), -I^-(x), -Ds(x) = +\})$
$Incompatible(\{I^-(x), -I^+(x), -Ds(x) = -\})$

[9] Only seven are required since two of the nine possible additions (when summing + and -) do not constrain the value of c at all).

$Incompatible(\{-I^+(x), -I^-(x), -Ds(x) = 0\})$

Of course, there is a great deal more to QPT than influence resolution, but we believe the entire mechanism should be specifiable using the Transition Calculus.

The system described here is still in an early stage of development and there are a number of improvements that could be made. For example, the present implementation would be likely to benefit from using an ATMS when checking inconsistency statements across many largely similar state sets. Another inefficiency in the current implementation is in the handling of sets of JEPD relations (for example the RCC relations or statements about the values of a variable in a large quantity space). In such cases, there will be a single positive literal in the state set description and many negative literals. State set descriptions may therefore become rather large and additional machinery or structuring may be required to obtain a satisfactory level of efficiency.

Transition Calculus includes a number of other features that we have not, so far, made use of. One such feature is the parallel composition of transition schema i.e. the ability to have events occurring concurrently. By making use of this, we hope to produce an enhanced simulator capable of modelling a wider range of systems than our present implementation.A further aspect of the calculus that we have not fully exploited is its nonmonotonic reasoning capabilities. We hope to exploit these in order to incorporate default behaviours that can be used to make predictions about system behaviour in the presence of incomplete information.

Acknowledgements

We are grateful to the Engineering and Science Research Council for supporting this work under grant number: GR/H 78955.

References

Allen, J. F. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM* 26(11):832–843.

Cohn, A. G.; Randell, D. A.; and Cui, Z. 1995. Taxonomies of logically defined qualitative spatial relations. *Int. J of Human-Computer Studies* 43:831–846.

Cui, Z.; Cohn, A. G.; and Randell, D. A. 1992a. Qualitative simulation based on a logical formalism of space and time. In *Proceedings AAAI-92*, 679–684. Menlo Park, California: AAAI Press.

Cui, Z.; Cohn, A. G.; and Randell, D. A. 1992b. Qualitative simulation based on a logic of space and time. In *QR-92*.

De Kleer, J., and Seely Brown, J. 1984. A qualitative physics based on confluences. *Artificial Intelligence* 24:7–83.

Forbus, K. D. 1990. The qualitative process engine. In Weld, D., and de Kleer, J., eds., *Readings in Qualitative Reasoning About Physical Systems*. Morgan Kaufmann.

Galton, A. 1995. Towards a qualitative theory of movement. In A Frank, W. K., ed., *Spatial Information Theory: a theoretical basis for GIS*, number 988 in Lecture Notes in Computer Science, 377–396. Berlin: Springer Verlag.

Gentzen, G. 1955. *Recherches sur La Deduction Logique*. Presses Universitaires de France.

Gooday, J. M., and Galton, A. P. 1996. The transition calculus: A high-level formalism for reasoning about action and change. *Journal of Theoretical and Experimental Artificial Intelligence*. To appear.

Gooday, J. M. 1994. *A transition-based approach to reasoning about action and change*. Ph.D. Dissertation, Department of Computer Science, University of Exeter, Exeter EX4 4PT, England.

Gotts, N. M. 1994. How far can we 'C'? defining a 'doughnut' using connection alone. In Doyle, J.; Sandewall, E.; and Torasso, P., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the 4th International Conference (KR94)*. Morgan Kaufmann.

Kuipers, B. 1994. *Qualitative Reasoning*. Cambridge, MA.: MIT Press.

Randell, D. A.; Cui, Z.; and Cohn, A. G. 1992. A spatial logic based on regions and connection. In *Proc. 3rd Int. Conf. on Knowledge Representation and Reasoning*, 165–176. San Mateo: Morgan Kaufmann.

Williams, B. C. 1991. A theory of interactions: Unifying qualitative and quantitative algebraic reasoning. *Artificial Intelligence* 39-94:51.