Chapter 5

Telephone domain

We perform complex commonsense reasoning about the effects of actions whenever we use a telephone. The behavior of an agent who uses a phone is highly context-sensitive. In this chapter, we first describe the telephone domain (Section 5.1) and later show an implementation of it (Section 5.2).

5.1 Description of the domain

Imagine a domain with only one agent. The agent should be aware of the answer he receives after dialing. If the agent hears *busy* or *not_at_home* answer, he can not talk. On the other hand, if he hears *hello* answer, he can talk. We assume that an agent can hear only one sound at a time. We model the domain with only one agent, but it can be extended to more than one. In addition, more actions and fluents can be added to make the domain more complete. Assume that, after the agent picks up the phone, he always hears *ready_to_dial* tone. In reality, if the phone is disconnected, the agent may hear *nothing* or, if the phone is connected to the Internet via modem, the agent may hear *busy* signal. We are simplifying the situation, but that kind of modifications are possible. We suppose that the agent has only one telephone number at hand and it is not a wrong number. The objects of the domain are the receiver of the phone and the telephone number, but another objects, such as cable and address book, can be added.

5.2 Implementation

In this section we present a representation of the telephone domain in ASP. We use Smodels to run the program.

```
% lparse --true-negation history1.sm | smodels 0| mkatoms

% HISTORY 1

% One agent picks up the receiver of a phone and hears a "Ready
% to dial signal".
% He dials the phone number and
% hears a "Hello" answer from the other side.

% QUESTION 1
% Can the agent talk?

% ANSWER 1
% yes

% HISTORY 2

% One agent picks up the receiver of a phone and hears "Nothing".
% QUESTION 2

% Can the agent talk?


% ANSWER 2

% no




% HISTORY 3


% One agent picks up the receiver of a phone and hears a "Ready

% to dial signal".

% He dials the phone number and

% hears a "Hello" answer from the other side. Immediately after that,

% he puts down the receiver.


% QUESTION 3

% Can the agent talk?


% ANSWER 3

% no


%%%%%%%%%%%%%%%%%%%%% DOMAIN DEPENDENT PART %%%%%%%%%%%%%%%%%%%%


#const n=2.
```

```
% Only one agent performs the actions


agent(agent1).

#domain agent(A;A1;A2).


% The receiver of the phone from where the agent calls


receiver(receiver1).

#domain receiver(R;R1;R2).


% The agent is going to call the telephone number


telephone_number(telephone_number1).

#domain telephone_number(N;N1;N2).


% The receiver and the telephone number are objects


object(X) :- receiver(X).

object(X) :- telephone_number(X).


#domain time(T;T1;T2).


% DOMAIN


% means that the phone on the opposite side is occupied
```

```
sound(busy).

% means that the agent on the opposite side is ready to talk

sound(hello).

% means that the agent on the opposite side is not available

sound(not_at_home).

% means that the agent is ready to dial

sound(ready_to_dial).

#domain sound(S;S1;S2).

% the agent is holding the receiver

fluent(holding(A,R)).

% the agent is hearing a sound

fluent(hearing(A,S)).

% the agent can talk
```

```
fluent(can_talk(A)).


% the agent can not talk


fluent(can_not_talk(A)).


#domain fluent(FL;FL1;FL2).


% the agent picks up the receiver


action(pickup(A,R)).


% the agent puts down the receiver


action(putdown(A,R)).


% the agent dials a number


action(dial(A,N)).
#domain action(ACT1;ACT2;ACT3).



% STATIC CAUSAL LAWS


% hearing is a function
```

```
-h(hearing(A,S1),T) :- h(hearing(A,S2),T),
                       S1 != S2.


% if the agent hears "hello", he can talk

h(can_talk(A),T) :- h(hearing(A,hello),T).

% if the agent hears hello, it is not true that he can not talk

-h(can_not_talk(A),T) :- h(hearing(A,hello),T).

% if the agent hears "not_at_home", he can not talk

h(can_not_talk(A),T) :- h(hearing(A,not_at_home),T).

% if the agent hears "not_at_home", it is not true that he can talk

-h(can_talk(A),T) :- h(hearing(A,not_at_home),T).

% if the agent hears "busy", he can not talk

h(can_not_talk(A),T) :- h(hearing(A,busy),T).

% if the agent hears "busy", it is not true that he can talk

-h(can_talk(A),T) :- h(hearing(A,busy),T).
```

```
% if the agent is not holding the phone, he can not hear anything

-h(hearing(A,S),T) :- -h(holding(A,R),T).

% if the agent is not holding the phone, he can not talk

h(can_not_talk(A),T) :- -h(holding(A,R),T).



% DYNAMIC CAUSAL LAWS and CONSTRAINTS

% when the agent picks up the phone, he can not talk, he needs to dial first

h(can_not_talk(A),T) :- o(pickup(A,R),T).

% when the agent picks up the phone, it is not true that he can talk,
% he needs to dial first

-h(can_talk(A),T) :- o(pickup(A,R),T).

% After the agent picks up the receiver,he is holding it

h(holding(A,R),T+1) :- o(pickup(A,R),T).

% After the agent picks up the receiver, the hears "ready_to_dial" tone
```

```
h(hearing(A,ready_to_dial),T+1) :- o(pickup(A,R),T).

% The agent can not pick up the receiver, if he is already holding it

:- o(pickup(A,R),T),
   h(holding(A,R),T).

% After the agent puts down the receiver, he is not holding it anymore

-h(holding(A,R),T+1) :- o(putdown(A,R),T).

% The agent can not putdown the receiver, if he is not holding it

:- o(putdown(A,R),T),
   -h(holding(A,R),T).

% If the agent hears "ready_to_dial", he can dial

o(dial(A,N),T) :- h(hearing(A,ready_to_dial),T).

% The agent can not dial if he does not hear "ready_to_dial"

:- o(dial(A,N),T),
   -h(hearing(A,ready_to_dial),T).
```

```
% After the agent dials the number, he can not hear "ready_to_dial" anymore

-h(hearing(A,ready_to_dial),T+1) :- o(dial(A,N),T).



% CHOICE RULES

% After dialing, the agent can hear only one out of three possible sounds

1{h(hearing(A,not_at_home),T+1), h(hearing(A,busy),T+1),
 h(hearing(A,hello),T+1)}1 :- o(dial(A,N),T).

% After dialing, the agent can either talk or not,
% depending on the answer received

1{h(can_talk(A),T+1), h(can_not_talk(A),T+1)}1 :- o(dial(A,N),T).

% DESCRIPTION OF THE INITIAL STATE (OBSERVATIONS)

% Initially, the agent picks up the receiver of the phone

o(pickup(agent1,receiver1),0).

% ENUMERATION RULES

% Initially, if there is no evidence that a fluent does not hold, then it holds
```

```
h(FL,0) :- not -h(FL,0).

% Initially, if there is no evidence that a fluent holds, then it does not hold

-h(FL,0) :- not h(FL,0).

%%%%%%%%%%%%%%%%%%%%%%%%% DOMAIN INDEPENDENT PART %%%%%%%%%%%%

% DEFINING TIME

time(0..n).

% INERTIA RULE

% Normally fluents stay as they are

h(FL,T+1) :- T<n, h(FL,T), not -h(FL,T+1).

-h(FL,T+1) :- T<n, -h(FL,T), not h(FL,T+1).


% CUSTOMIZE THE OUTPUT

true(FL,T) :- h(FL,T).
```

```
show true(FL,T).


hide h(FL,T).
hide fluent(FL).
hide object(X).
hide time(X).
hide agent(X).
hide action(X).
hide receiver(X).
hide telephone_number(X).
hide sound(X).
```

We obtained the answers that we expected. The same knowledge about the telephone domain is represented using Discrete Event Calculus in [Mueller, 2006]. We list the axioms used there in the next section and we let the reader to compare the two alternative representations.

5.3 Telephone domain represented by Mueller

If an agent picks up an idle phone, the phone will have a dial tone and will no longer be idle:

$$HoldsAt(Idle(p), t) \Longrightarrow$$
$$Initiates(PickUp(a, p), DialTone(p), t)$$

$$HoldsAt(Idle(p), t) \Longrightarrow$$
$$Terminates(PickUp(a, p), Idle(p), t)$$

After picking up a phone, an agent may decide not to place a call. If an agent sets down a phone with a dial tone, the phone will be idle and will no longer have a dial tone:

$$HoldsAt(DialTone(p), t) \implies Initiates(SetDown(a, p), Idle(p), t)$$

$$HoldsAt(DialTone(p), t) \implies Terminates(SetDown(a, p), DialTone(p), t)$$

When phone $p1$ has a dial tone and an agent dials phone $p2$ from $p1$, what happens depends on the state of $p2$. If $p2$ is idle, then $p1$ will be ringing $p2$, $p1$ will no longer have a dial tone, and $p2$ will no longer be idle:

$$\begin{gathered} HoldsAt(DialTone(p1), t) \wedge HoldsAt(Idle(p2), t) \implies \\ Initiates(Dial(a, p1, p2), Ringing(p1, p2), t) \end{gathered}$$

$$\begin{gathered} HoldsAt(DialTone(p1), t) \wedge HoldsAt(Idle(p2), t) \implies \\ Terminates(Dial(a, p1, p2), DialTone(p1), t) \end{gathered}$$

$$\begin{gathered} HoldsAt(DialTone(p1), t) \wedge HoldsAt(Idle(p2), t) \implies \\ Terminates(Dial(a, p1, p2), Idle(p2), t) \end{gathered}$$

If $p2$ is not idle, then $p1$ will have a busy signal:

$$\begin{gathered} HoldsAt(DialTone(p1), t) \wedge \neg HoldsAt(Idle(p2), t) \implies \\ Initiates(Dial(a, p1, p2), BusySignal(p1), t) \end{gathered}$$

$$HoldsAt(DialTone(p1),t) \land \neg HoldsAt(Idle(p2),t) \Longrightarrow$$
$$Terminates(Dial(a,p1,p2),DialTone(p1),t)$$

If an agent sets down a phone with a busy signal, then the phone will be idle and will no longer have a busy signal:

$$HoldsAt(BusySignal(p),t) \Longrightarrow$$
$$Initiates(SetDown(a,p),Idle(p),t)$$

$$HoldsAt(BusySignal(p),t) \Longrightarrow$$
$$Terminates(SetDown(a,p),BusySignal(p),t)$$

A call may go unanswered. If phone $p1$ is ringing phone $p2$ and an agent sets down $p1$, then $p1$ will be idle, $p2$ will be idle, and $p1$ will no longer be ringing $p2$:

$$HoldsAt(Ringing(p1,p2),t) \Longrightarrow$$
$$Initiates(SetDown(a,p1),Idle(p1),t)$$

$$HoldsAt(Ringing(p1,p2),t) \Longrightarrow$$
$$Initiates(SetDown(a,p1),Idle(p2),t)$$

$$HoldsAt(Ringing(p1,p2),t) \Longrightarrow$$
$$Terminates(SetDown(a,p1),Ringing(p1,p2),t)$$

A call may be answered. If phone $p1$ is ringing phone $p2$ and an agent picks up $p2$, then $p1$ will be connected to $p2$ and $p1$ will no longer be ringing $p2$:

$$HoldsAt(Ringing(p1,p2),t) \Longrightarrow$$
$$Initiates(PickUp(a,p2),Connected(p1,p2),t)$$

$$HoldsAt(Ringing(p1,p2),t) \Longrightarrow$$
$$Terminates(PickUp(a,p2),Ringing(p1,p2),t)$$

A call may be completed. If phone $p1$ is connected to phone $p2$ and an agent sets down $p1$, then $p1$ will be idle, $p2$ will be disconnected, and $p1$ will no longer be connected to $p2$:

$$HoldsAt(Connected(p1,p2),t) \Longrightarrow$$
$$Initiates(SetDown(a,p1),Idle(p1),t)$$

$$HoldsAt(Connected(p1,p2),t) \Longrightarrow$$
$$Initiates(SetDown(a,p1),Disconnected(p2),t)$$

$$HoldsAt(Connected(p1,p2),t) \Longrightarrow$$
$$Terminates(SetDown(a,p1),Connected(p1,p2),t)$$

If phone $p1$ is connected to phone $p2$ and an agent sets down $p2$, then $p2$ will be idle, $p1$ will be disconnected, and $p1$ will no longer be connected tp $p2$:

$$HoldsAt(Connected(p1,p2),t) \Longrightarrow$$
$$Terminates(SetDown(a,p2),Idle(p2),t)$$

$$HoldsAt(Connected(p1, p2), t) \Longrightarrow$$
$$Initiates(SetDown(a, p2), Disconnected(p1), t)$$

$$HoldsAt(Connected(p1, p2), t) \Longrightarrow$$
$$Terminates(SetDown(a, p2), Connected(p1, p2), t)$$

If an agent sets down a phone that is disconnected, then the phone will be idle and the phone will no longer be disconnected:

$$HoldsAt(Disconnected(p), t) \Longrightarrow$$
$$Initiates(SetDown(a, p), Idle(p), t)$$

$$HoldsAt(Disconnected(p), t) \Longrightarrow$$
$$Terminates(SetDown(a, p), Disconnected(p), t)$$

At timepoint 0, all phones are idle, no phone have a dial tone or busy signal, no phones are ringing other phones, no phones are connected to other phones, and no phones are disconnected:

$$HoldsAt(Idle(p), 0)$$
$$\neg HoldsAt(DialTone(p), 0)$$
$$\neg HoldsAt(BusySignal(p), 0)$$
$$\neg HoldsAt(Ringing(p1, p2), 0)$$
$$\neg HoldsAt(Connected(p1, p2), 0)$$

$$\neg HoldsAt(Disconnected(p), 0)$$

Fluents are never released from the commonsense law of inertia:

$$\neg ReleasedAt(f, t)$$

One agent picks up the phone and dials another agent, and then the other agent answers:

$$Happens(PickUp(Agent1, Phone1), 0)$$
$$Happens(Dial(Agent1, Phone1, Phone2), 1)$$
$$Happens(PickUp(Agent2, Phone2), 2)$$

The two agents will be connected. The reader can find the proof of the functionality of the previous rules in [Mueller, 2006].

5.4 Closing remarks

In this chapter we presented the telephone domain in both ASP and in Discrete Event Calculus. Even though in this case, both representations give similar results, we recommend the use of ASP for commonsense knowledge representation and reasoning, because it is less complicated, but at the same time it has strong theoretical foundations. Using Smodels, we realized that it was sufficient to express all rules of our description. In Chapter 6, we mention more about the work of Erik Mueller.