

Relational Approach to Knowledge Engineering for POMDP-based Assistance Systems as a Translation of a Psychological Model

Marek Grzes^{a,*}, Jesse Hoey^a, Shehroz S. Khan^a, Alex Mihailidis^b, Stephen Czarnuch^b, Dan Jackson^c, Andrew Monk^d

^a*School of Computer Science, University of Waterloo, Canada*

^b*Department of Occupational Science and Occupational Therapy, University of Toronto, Canada*

^c*School of Computing Science, Newcastle University, United Kingdom*

^d*Department of Psychology, University of York, United Kingdom*

Abstract

Assistive systems for persons with cognitive disabilities (e.g. dementia) are difficult to build due to the wide range of different approaches people can take to accomplishing the same task, and the significant uncertainties that arise from both the unpredictability of client's behaviours and from noise in sensor readings. Partially observable Markov decision process (POMDP) models have been used successfully as the reasoning engine behind such assistive systems for small multi-step tasks such as hand washing. POMDP models are a powerful, yet flexible framework for modelling assistance that can deal with uncertainty and utility. Unfortunately, POMDPs usually require a very labour intensive, manual procedure for their definition and construction. Our previous work has described a knowledge driven method for automatically generating POMDP activity recognition and context sensitive prompting systems for complex tasks. We call the resulting POMDP a SNAP (SyNdetic Assistance Process). The spreadsheet-like result of the analysis does not correspond to the POMDP model directly and the translation to a formal POMDP representation is required. To date, this translation had to be performed manually by a trained POMDP expert. In this paper, we formalise and automate this translation process using a probabilistic relational model (PRM) encoded in a relational database. The database encodes the relational skeleton of the PRM, and includes the goals, action preconditions, environment states, cognitive model, client and system actions (i.e., the outcome of the SNAP analysis), as well as relevant sensor models. The database is easy to approach for someone who is not an expert in POMDPs, allowing them to fill in the necessary details of a task using a simple and intuitive procedure. The database, when filled, implicitly defines a ground instance of the relational skeleton, which we extract using an automated procedure, thus generating a POMDP model of the assistance task. A strength of the database is that it allows constraints to be specified, such that we can verify the POMDP model is, indeed, valid for the task given the analysis. We demonstrate the method by eliciting three assistance tasks from non-experts: handwashing, and toothbrushing for elderly persons with dementia, and on a factory assembly task for persons with a cognitive disability. We validate the resulting POMDP models using case-based simulations to show that they are reasonable for the domains. We also show a complete case study of a designer specifying one database, including an evaluation in a real-life experiment with a human actor.

Keywords: dynamic Bayesian networks, knowledge engineering, probabilistic planning, POMDPs, reinforcement learning, COACH, assistance systems, handwashing, prompting system, SQL, relational database

1. Introduction

Quality of life (QOL) of persons with a cognitive disability (e.g. dementia, developmental disabilities) is increased significantly if they can engage in 'normal' routines in their own homes, workplaces, and communities. However,

*Corresponding author

Email addresses: mgrzes@cs.uwaterloo.ca (Marek Grzes), jhoey@cs.uwaterloo.ca (Jesse Hoey), s255khan@cs.uwaterloo.ca (Shehroz S. Khan), alex.mihailidis@utoronto.ca (Alex Mihailidis), stephen.czarnuch@utoronto.ca (Stephen Czarnuch), d.g.jackson@newcastle.ac.uk (Dan Jackson), a.monk@psych.york.ac.uk (Andrew Monk)

1
2
3 they generally require some assistance in order to do so. For example, difficulties performing common activities
4 such as self-care or work-related tasks, may trigger the need for personal assistance or relocation to residential care
5 settings [1]. Moreover, it is associated with diminished QOL, poor self-esteem, anxiety, and social isolation for the
6 person and their caregiver [2].

7
8 Technology to support people in their need to live independently is currently available in the form of personal and
9 social alarms and environmental adaptations and aids. Looking to the future, we can imagine intelligent, pervasive
10 computing technologies using sensors and effectors that help with more difficult cognitive problems in planning,
11 sequencing and attention. A key problem in the construction of such intelligent technologies is the automatic analysis
12 of people's behaviours from sensory data. Activities need to be recognized and, by incorporating domain specific
13 expert knowledge, reasonable conclusions have to be drawn which ultimately enables the environment to perform
14 appropriate actions through a set of actuators. In the example of assisting people with dementia, the smart environment
15 would prompt (i.e., issue a voice or video prompt) whenever the clients get stuck in their activities of daily living.

16 The technical challenges of developing useful prompts and a sensing and modelling system that allows them
17 to be delivered only at the appropriate time have been tackled through the use of advanced planning and decision
18 making approaches. One of the more sophisticated of these types of systems is the COACH [3]. COACH uses
19 computer vision to monitor the progress of a person with dementia washing their hands and prompts only when
20 necessary. COACH uses a partially observable Markov decision process (POMDP), a temporal probabilistic model
21 that represents a decision making process based on environmental observations. The COACH model is flexible in
22 that it can be applied to different tasks [4]. However, each new task requires substantial re-engineering and re-design
23 to produce a working assistance system, which currently requires massive expert knowledge for generalization and
24 broader applicability to different tasks. An automatic generation of such prompting systems would substantially
25 reduce the manual efforts necessary for creating assistance systems, which are tailored to specific situations and tasks,
26 and environments. In general, the use of *a-priori* knowledge in the design of assistance systems is a key unsolved
27 research question. Researchers have looked at specifying and using ontologies [5], information from the Internet [6],
28 logical knowledge bases [5, 7], and programming interfaces for context aware human-computer interaction [8].

29
30 In our previous work, we have developed a knowledge driven method for automatically generating POMDP ac-
31 tivity recognition and context sensitive prompting systems [9]. The approach starts with a description of a task and
32 the environment in which it is to be carried out that is relatively easy to generate. Interaction Unit (IU) analysis [10],
33 a psychologically motivated method for transcoding interactions relevant for fulfilling a certain task, is used for ob-
34 taining a formalized, i.e., machine interpretable task description. We call the resulting model a SyNdetic Assistance
35 Process (SNAP). However, the current system uses an ad-hoc method for transcoding the IU analysis into the POMDP.
36 While each of the factors are well defined, fairly detailed and manual specification is required to enable the translation.

37 The long-term goal of the approach presented in this paper is to allow end-users, such as health professionals,
38 caregivers, and family members, to specify and develop their own context sensitive prompting systems for their needs
39 as they arise. This paper describes a step in this direction by proposing a probabilistic relational model (PRM)[11]
40 defined as a relational database that encodes a domain independent relational dynamic model and serves to mediate
41 the translation between the IU analysis and the POMDP specification. The PRM encodes the constraints required
42 by the POMDP in such a way that, once specified, the database can be used to generate a POMDP specification
43 automatically that is guaranteed to be valid (according to the SNAP model). The PRM serves as a schema that can
44 be instantiated for a particular task using a simple and intuitive specification method. The probabilistic dependencies
45 in the PRM are boiled down to a small set of parameters that additionally need to be specified to produce a working
46 POMDP-based assistance system. We show how the method requires little prior knowledge of POMDPs, and how it
47 makes specification of relatively complex tasks a matter of a few hours of work for a single coder.

48
49 The remainder of this paper is structured as follows. First, we give an overview of the basic building blocks:
50 POMDPs, the IU analysis, knowledge engineering, and probabilistic relational models (PRMs). Then, Section 3
51 describes the specific PRM and relational database that we use, and shows how the database can be leveraged in the
52 translation of the IU analysis to a POMDP planning system. In Section 4, we show a case study that explains step-
53 by-step the design process of an example prompting system. Section 5 shows how the method can be applied to three
54 tasks, including the real-life simulation with a human actor, and then the paper concludes.

55 This paper is describing assistive systems to help persons with a cognitive disability. Throughout the paper, we
56 will refer to the person requiring assistance as the *client*, and to any person providing this assistance as the *caregiver*.
57 A third person involved is the *designer*, who will be the one using the system we describe in this paper to create the
58

assistive technology. Thus, our primary target user in this paper is the *designer*.

2. Overview of Core Concepts

We start by introducing the concept of POMDPs (Section 2.1) which is the core mathematical model and represents the final outcome of our modelling task, i.e., the actual machine-readable specification of the prompting system. Next, we introduce a method that is used in order to create the initial SNAP specification for a new task as a spreadsheet-like document (Section 2.2). Section 2.3 then reviews key concepts of knowledge engineering, and motivates the primary objective of this paper: to provide a link between the task analysis of Section 2.2 and the POMDP model. Finally, Section 2.4 overviews probabilistic relational models (PRMs). We use a PRM as our primary abstraction of the POMDP model, enabling designers to specify POMDP models at a level of abstraction that is appropriate.

2.1. Partially observable Markov decision processes

A POMDP is a probabilistic temporal model of a system interacting with its environment [12, 13], and is described by (1) a finite set of state variables, the cross product of which gives the state space, S ; (2) a set of observation variables, O (the outputs of some sensors); (3) a set of system actions, A ; (4) a reward function, $R(s, a, s')$, giving the relative utility of transiting from state s to s' under action a ; (5) a stochastic transition model $Pr : S \times A \rightarrow \Delta S$ (a mapping from states and actions to distributions over states), with $Pr(s'|s, a)$ denoting the probability of moving from state s to s' when action a is taken; and (6) a stochastic observation model with $Pr(o|s)$ denoting the probability of making observation o while the system is in state s . Figure 1(a) shows a POMDP as a Dynamic Bayesian network (DBN) with actions and rewards, where arrows are interpretable as causal links between variables.

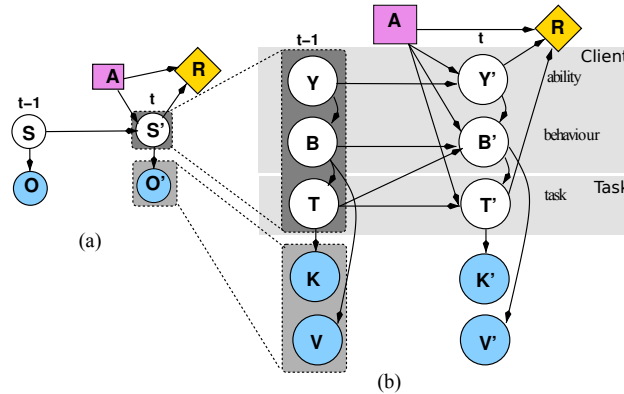


Figure 1: Two time slices of (a) a general POMDP; (b) a factored POMDP for interactions with assistive technology. T is the set of task variables, B represents behaviours of the client, and Y client’s abilities. B and Y constitute the model of the client. A is the action of the system, R the reward, and K and V are observations/sensors for task variables and client’s behaviours correspondingly. Primed variables represent the next time step of the dynamic model and arrows indicate probabilistic dependence.

A POMDP for personal assistance breaks the state space down into three key factors as shown in Figure 1(b): states describing elements of the functional task in the real world, T , e.g. whether the water has been boiled or not (the ‘task factor model’), states capturing the client’s cognitive capacities, Y , e.g., to remember what they are supposed to do next (the ‘ability factor model’), and states capturing an inferred history of what the client has actually done since the last update, B , e.g. fill the kettle (the ‘behaviour factor model’). We use the word ‘behaviour’ here to describe actions of the client to distinguish them from actions of the system (i.e., prompts to the client). A preliminary version of this model was explored and used in different contexts in [14, 15, 3]. As we will see, these three factors relate to the psychological analysis, and keeping them separate will ease the translation between the two.

The system’s actions are various prompts or memory aids the system can use to help a client remember things in the task. For persons with dementia, simple audio prompts are often very effective, and are used in the COACH system along with video demonstrations [15, 3]. Finally, the observations, O , are any information from sensors in the environment that can give evidence to the system about the client’s behaviours or the task, divided into the states of sensors relevant to the task environment (K) and states of sensors relevant to client behaviour (V). Whereas the

1
2
3 first type of sensors monitors, for example, movements of objects in the environment, the latter corresponds to virtual
4 sensors that, for example, monitor the activities the acting person pursues.

5 Once specified, the POMDP can be solved with a planner in order to obtain a policy of action, i.e., to determine
6 in which states of the environment prompts should be issued. Thus our work in this paper does not focus on solving
7 POMDPs but rather on methodologies for specifying POMDPs.
8

9
10 **2.2. Specifying the task: Interaction Unit Analysis**

11 Task analysis has a long history in Human Factors [16] where this approach is typically used to help define and
12 break-down ‘activities of daily living’ (ADL) – i.e. activities that include self-care tasks, household duties, and
13 personal management such as paying bills. The emphasis in task analysis is on describing the actions taken by a client
14 and the intentions (goals and sub-goals) that give rise to those actions. There has been less emphasis on how actions
15 are driven by the current state or changes in the environment. Syndetic modeling [17] remedies this omission by
16 describing the conjunction of cognitive and environmental precursors for each action. Modelling both cognitive and
17 environmental mechanisms at the level of individual actions turns out to be much more efficient than building separate
18 cognitive and environmental models [10].

19 The task analysis technique [18] breaks a task down into a set of goals, states, abilities and behaviours, and defines
20 a hierarchy of tasks that can be mapped to a POMDP, a policy for which will be a situated prompting system for a
21 particular task [9]. The technique involves an experimenter video-taping a person being assisted during the task, and
22 then transcribing and analysing the video. The end-result is an Interaction Unit (IU) analysis that uncovers the states
23 and goals of the task, the client’s cognitive abilities, and the client’s actions. A simplified example for the first step in
24 tea-making (getting out the cup and putting in a tea-bag) is shown in Table 1. The rows in the table show a sequence of
25 steps, with the client’s current goals, the current state of the environment, the abilities that are necessary to complete
26 the necessary step, and the behaviour that is called for. The abilities are broken down into ability to *recall* what they
27 are doing, to *recognise* necessary objects like the kettle, and to perceive *affordances* of the environment.
28

29
30

IU	Goals	Task States	Abilities	Behaviours
1	Final	cup empty on tray, box closed	Rn cup on tray, Rl step	No Action
2	Final, cup TB	cup empty on tray, box closed	Af cup on tray WS	Move cup tray→WS
3	Final, cup TB	cup empty on WS, box closed	Rl box contains TB, Af box closed	Alter box to open
4	Final, cup TB	cup empty on WS, box open	Af TB in box cup	Move TB box→cup
5	Final	cup tb on WS, box open	Af box open	Alter box to closed
6	Final	cup tb on WS, box closed		

31
32
33
34
35
36
37 Table 1: IU analysis of the first step in tea making. The first column provides an index of a specific interaction unit (a row) in the IU table. The
38 second column contains the goal of the client in a specific row, whereas the third column defines the set of task states which are relevant in the row.
39 The fourth column lists abilities which are necessary for the behaviour shown in the fifth column to happen. Types of behaviours are distinguished
40 by the prefix of their names, e.g., Rn=recognition, Rl=Recall, Af=Affordance. The remaining shorthand notation is: tb=teabag, ws=work surface.

41 A second stage of analysis involves proposing a set of sensors and actuators that can be retrofitted to the client’s
42 environment for the particular task, and providing a specification of the sensors that consists of three elements: (1)
43 a name for each sensor and the values it can take on (e.g. on/off); (2) a mapping from sensors to the states and
44 behaviours in the IU analysis showing the evidentiary relationships, and (3) measurements of each sensor’s reliability
45 at detecting the states/behaviours it is related to in the mapping.
46

47 The IU analysis (e.g. Table 1), along with the specification of sensors which are part of SNAP can be converted to
48 a POMDP model by factoring the state space as shown in Figure 1(b). The method is described in detail in [9]; here we
49 give a brief overview. The *task* variables are a characterisation of the domain in terms of a set of high-level variables,
50 and correspond to the entries in the task states column in Table 1. For example, in the first step of tea making, these
51 include the box condition (open, closed) and the cup contents (empty or with teabag). The task states are changed
52 by the client’s *behaviour*, *B*, a single variable with values for each behaviour in Table 1. For the first IU group in
53 tea making, these include opening/closing the box, moving the teabag to the cup, and doing nothing or something
54 unrelated (these last two behaviours are always present). The clients’ *abilities* are their cognitive states, and model the
55 ability of the client to recall (Rl), recognise (Rn) and perceive affordances (Af). For the first IU group, these include
56 the ability to recognise the tea box and to perceive the affordance of moving the teabag to the cup. Example DBN
57

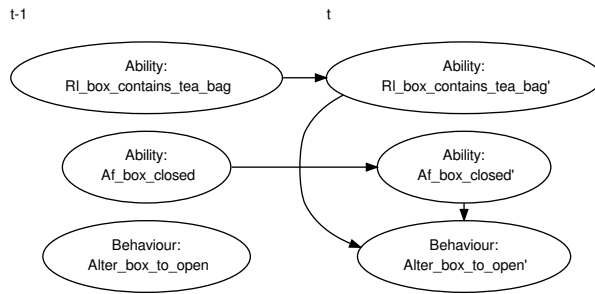


Figure 2: Example probabilistic dependencies read from the IU table specified in Table 1. The example corresponds to row 3 in Table 1 that specifies that behaviour `Alter_box_to_open` requires abilities `RI_box_contains_tea_bag` and `Af_box_closed`.

relationships read from Table 1 are shown in Figure 2. The example corresponds to row 3 in Table 1 that specifies that behaviour `Alter_box_to_open` requires abilities `RI_box_contains_tea_bag` and `Af_box_closed`.

The system actions are prompts that can be given to help the client regain a lost ability. We define one system action for each necessary ability in the task. The actions correspond to a prompt or signal that will help the client with this particular ability, if missing. System actions (prompts) are automatically derived from the IU analysis because abilities of different types are specified by the person performing the analysis, and the system can then automatically generate one system action (prompt) for every ability.

Some of the required dynamics (i.e., behaviour relevance) and initial state are produced directly from the IU analysis (Table 1), by looking across each row and associating state transitions between rows. We take this to be deterministic, as any uncertainty will be introduced by the client’s abilities (so we assume a perfectly able client is able to always successfully complete each step). Each prompting action improves its associated cognitive ability. For example, the ‘prompt recognition cup’ action (e.g a light shone on the cup) makes it more likely that the client can recognise the cup if they cannot do so already. The reward function specifies the goal states (in Table 1), and assigns a cost to each prompt, as client independence is paramount.

2.3. Requirements from Knowledge Engineering

For an extensive review of the challenges which knowledge engineering (KE) for planning faces, the reader is referred to [19] where authors collected a number of requirements that should be satisfied. In our work on the SNAP process, we found that these requirements can be, to a great extent, supported when one applies the relational database formalism to store and to process the domain model. For example, validation of the data is highly supported and the relational database language is structured, expressive, customizable and has a clear syntax and semantics.

2.4. Probabilistic Relational Models

In the application areas which are considered in this paper, planning problems are POMDPs. POMDPs can be seen as Dynamic Decision Networks (DDNs) [20]. In most POMDP planners, DDNs have a propositional representation, where the domain has a number of attributes, and attributes can take values from their corresponding domains. As long as this is satisfactory for planning purposes (i.e., when the POMDP has already been defined), the problem with designing methodologies and tools for engineering the definitions of such planning problems using propositional techniques is that the reuse of the model in new instances is not straightforward because the model ignores relations between various entities of the domain, and therefore a relational approach becomes useful. Statistical Relational Learning [21] makes relational specification of probabilistic models possible as it allows for defining different types of objects (classes) and their sets of attributes and relations between classes. The specific model we focus on here is the probabilistic relational model, or PRM [11].

Probabilistic Relational Models (PRM) define a template for a probability distribution over attributes of objects (or columns of tables in database terminology) that specifies a concrete probability distribution when grounded with specific data. Figure 3 shows the main components of the probabilistic model specified using the PRM. The first element is the relational schemata which can be formalised as a specification of types of objects (classes), their attributes, and relations between objects of specific types. The two additional components are: for each attribute the set of parents

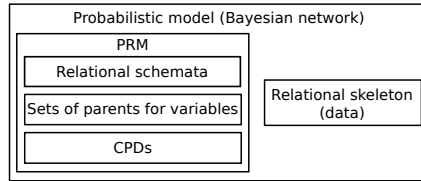


Figure 3: The probabilistic model and its components when specified as a PRM. The relational schemata represents classes of objects, attributes/slots of classes, and relations between classes via reference slots. Sets of parents of variables specify which attributes influence other attributes in their conditional probability distributions. Conditional probability distributions are also required for every random variable. The relational skeleton provides specific objects for which the relational model allows obtaining a probability distribution over unobserved attributes of the skeleton.

the attribute probabilistically depends on (and the aggregating method when the probability distribution depends on a variable number of objects), and the corresponding conditional probability distributions/tables (CPDs/CPTs).

A Bayesian network can be seen as a PRM with one class and no relations. The primary advantage of the PRM is that it distinguishes more than one class, implying that relationships between classes can be identified and then conditional probability tables (CPTs) can be shared among many objects. In particular, aggregating rules (details below) can facilitate CPTs which can depend on a changeable number of objects. Overall, models can be represented more compactly. The relational specification of a POMDP also requires the reward function, which does not appear in the original definition of the PRM in [11]. Since any Bayesian network can be implicitly interpreted as a decision network when utilities are provided externally, or have explicit utility nodes [22], utilities are a natural extension to the original PRM.

The relational schemata of the PRM can be represented directly as a standard relational database where tables and derived tables determined by SQL queries define objects, and columns in tables define attributes. Relationships between objects are modelled as primary/foreign key constraints or by additional tables which model relationships/links. The key property of PRMs that we exploit in our model is that properties of objects can be represented either as attributes of classes/tables or as separate objects and connected with the main object using classes/tables which model relationships/links. Since conditional probability tables (CPTs) are parametrised, the CPT for a given type of attribute is the same in all possible objects, i.e., one CPT is reused for multiple objects. Aggregating functions of a variable number of objects are often used to achieve this property [22, 23, 24].

3. Relational Database Model of SNAP

Our main emphasis is that a non-technical designer should be able to define the prompting system (the AI planning problem) easily and with minimal technical knowledge of probabilistic planning. The approach we are proposing in this paper is to provide a probabilistic relational schema (as a relational database) which can be populated by the designer using standard database tools such as forms or web interfaces, and then grounding the PRM to a POMDP specification using a *generator* software, which implements parts of the PRM not stored in the database (such as aggregation operators). In this section, we first give the PRM schema definition, followed by the precise specification of the aggregation operators that allow the probability distributions in the POMDP to be generated.

In the following, we use sans-serif font for class names, *Capitalised Italics* for attributes and *lower-case italics* for objects or attribute values. Set of attributes or values are in ***boldface italics***. The attributes, referred to also as slots, A , of a class, X are denoted $X.A$. A ***slot chain*** is an attribute of a class that is an external reference to another class, and so consists of a set of objects, as $X.A$.

3.1. Relational Schemata

The relational schema is composed of a set of classes, one for each variable (including observations and actions) in the POMDP shown in Figure 1(b). The class name is the variable name. The attributes/slots of each class include (at least) an object *Name* (e.g. an ability object with $Name = rn_cup_on_tray$) and a typed *Value*. The value of each object is the value of the corresponding variable (or action) in the POMDP. We assume here without loss of generality that all variables except the *Task* are Boolean. Any N -valued variable can always be converted to Boolean by splitting it into N Boolean variables (thereby allowing concurrent values). For example, in our previous work, we assume the

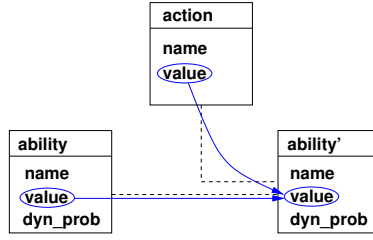


Figure 4: Relational schema and dependency structure for PRM for the client ability dynamics, $P(y'|y, a)$. Client's ability depends on the system's action and on the value of at least the same ability in the previous time step.

client *Behaviour* is a single variable, and thus the client can only perform one behaviour (out of N_B total) at a time. More generally, we can split this into N_B Boolean variables and allow the client to be performing multiple behaviours simultaneously (e.g. talking on the phone and pouring water). Similarly for system actions, we can allow concurrency or not.

In the following, we will examine the five different CPTs for the POMDP in Figure 1(b), showing the relevant piece of the PRM, the dependency structure, and we will show how the CPT is derived using aggregation operators on the PRM.

3.1.1. Client's Abilities Dynamics Model - Y'

Actions of our prompting systems depend on specific abilities that the client has to possess in order to complete specific steps of the task. In our relational model, it is assumed that the system can either (a) prompt for a specific ability, or (b) do nothing (a no-op action that is present in all instantiations of the model). The generator software creates one action for each ability in the domain. The prompt for an ability increases the chance that the client will regain that ability. For example, if the client lacks the ability to recognise the cup, the system may flash a light on the cup in order to focus client's attention on it.

Figure 4 shows the dependency structure and relational schema for the PRM for the client's ability dynamics, $P(Y'|Y, A)$. The abilities have an additional slot *Dyn_prob* that includes a set of four probabilities: $\{keep_prompt, gain_prompt, keep, gain\}$ that are used to define the CPT for the ability dynamics:

$$P(\text{Ability}'.Value | \text{Pa}(\text{Ability}'.Value))$$

Here, the parents of the ability are

$$\text{Pa}(\text{Ability}'.Value) = \{\gamma_1(\text{Ability}'.Action.Value), \gamma_2(\text{Ability}'.Ability.Value)\}$$

and $\gamma_1(\mathbf{A}) = \vee(\mathbf{A})$ is the aggregate (disjunction) of all actions that affect *Ability'.Value*: any action can have this effect. Similarly, $\gamma_2(\mathbf{Y}) = \wedge(\mathbf{Y})$ is the aggregate (conjunction) all previous abilities that affect *Ability'.Value* (usually this is only the same ability from the previous time-step). The CPT defines what happens to a particular client ability when an action is taken in a state where the client has a current set of abilities. If the action corresponds to that ability (e.g. is a prompt for it), then it will be in the set *Ability'.Action*, and thus γ_1 will be *true*. The dynamics of an ability is also conditioned on the previous abilities the client had, as given by the set *Ability'.Ability*. If all these necessary precondition abilities are present, then γ_2 will be *true*. The CPT is then defined as follows, where we use the shorthand Y' to denote *ability'.value*, and \mathbf{Y}, \mathbf{A} to denote *ability'.Ability*, *ability'.Action*.

$\gamma_1(\mathbf{A})$	$\gamma_2(\mathbf{Y})$	$P(Y' = true \mathbf{Y}, \mathbf{A})$
true	true	<i>ability'.keep_prompt</i>
true	false	<i>ability'.gain_prompt</i>
false	true	<i>ability'.keep</i>
false	false	<i>ability'.gain</i>

Suppose a prompt is given for *ability'*. Then, if the ability preconditions are satisfied, the client will have the *ability* in the next time step with probability *ability'.keep_prompt*. If, on the other hand, the ability preconditions are

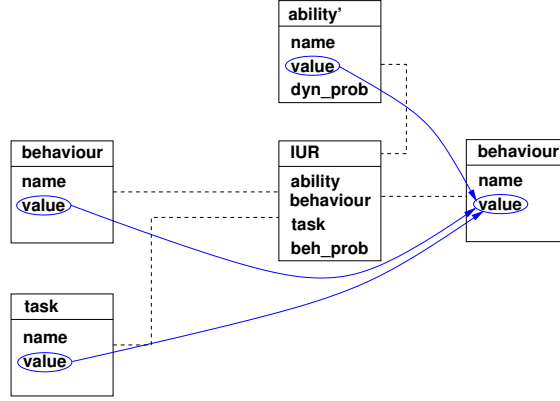


Figure 5: Relational schema and dependency structure for PRM for the client behaviour dynamics, $P(B'|B, Y', T)$. The client's behaviour depends on the previous behaviour, current ability and the previous task state - this dependency originates from Table 1.

not satisfied, this probability will be $ability'.gain_prompt$. The probabilities $ability'.keep$ and $ability'.gain$ are for the situation when the action is *not* related to the ability in question.

3.1.2. Client Behaviour Dynamics Model - B'

Figure 5 shows the dependency structure and relational schema for the PRM for the client behaviour dynamics, $P(B'|B, T, Y')$ (we leave off dependency on actions for clarity of presentation). In order to define the aggregation operators, we need to introduce a *link* class, IUR, that represents the IU table (e.g. Figure 1). An IUR object is a single row of the IU table, and has sets of abilities, behaviours, and task variables, along with a *Beh_prob* slot that gives the probability of a particular behaviour set occurring. We are seeking the CPT:

$$P(\text{Behaviour}'.\text{Value} | Pa(\text{Behaviour}'.\text{Value}))$$

where the parents of $\text{Behaviour}'.\text{Value}$ are an aggregate of $\text{Behaviour}'.\text{IUR}.\text{Ability}'$, $\text{Behaviour}'.\text{IUR}.\text{Task}$ and $\text{Behaviour}'.\text{IUR}.\text{Behaviour}'$. However, we cannot define a variable as the aggregate, and instead directly define the probability distribution of interest.

Denote the set of rows in the IU table as I . T , T' , B , B' , Y , and Y' are as specified in Figure 1(b). In order to compactly specify the CPT, we define the following Boolean aggregation functions:

1. $row_rel : I \times T \rightarrow \{0, 1\}$ is 1 for task states relevant in row, i , and 0 otherwise, and aggregates all task variables in $\text{Behaviour}'.\text{IUR}.\text{Task}$ as

$$row_rel(i) = \bigwedge (\text{Behaviour}'.row.i.\text{Task}.\text{Value}). \quad (1)$$

We write this as a function of the row, i , only: $row_rel(i)$ leaving the remaining variables implicit. The same shorthand is applied to the other functions.

2. $row_abil_rel : I \times Y' \rightarrow \{0, 1\}$ is 1 if all abilities on row i are present, and aggregates all ability variables in $\text{Behaviour}'.\text{IUR}.\text{Ability}'$ as

$$row_abil_rel(i) = \bigwedge (\text{Behaviour}'.row.i.\text{Ability}'.\text{Value}). \quad (2)$$

There may be multiple rows in the IU table with the same set of task states (i.e. there are two possible behaviours, possibly requiring different abilities, that are possible and relevant in a task state). The IUR class has an additional slot $beh_prob(i)$ that gives the probability that the behaviour on row i will take place. $beh_prob(i)$ must be a well defined probability distribution for each row, so if $behaviour(i)$ means the behaviour on row i , we must have

$$\forall_b \left(\sum_{i | behaviour(i)=b} beh_prob(i) = 1 \right) \quad (3)$$

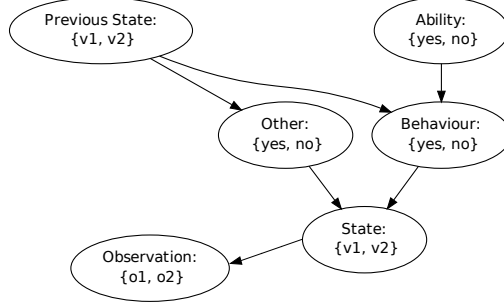


Figure 6: A simple Bayesian network that explains how regression is implemented in our current system. Behaviour other yields a state transition to all values of the state with a strong preference towards keeping all features with their existing values.

With the two aggregations defined above and this probability, we can define the CPT of a behaviour, B' , if the set of rows on which that behaviour appears is I_b , as

$$P(B' = true|Y', T) = \bigvee_{i \in I_b} \left(\begin{array}{l} row_abil_rel(i) \wedge \\ row_rel(i) \wedge \\ beh_prob(i) \end{array} \right) \quad (4)$$

This distribution is formed as a disjunction of the conjunction of aggregations because there is some non-zero probability of the behaviour for *any* task state that appears in $row_rel(i)$ where $i \in I_b$, and if the rows are not mutually exclusive, then the $beh_prob(i)$ should satisfy the condition (3). To evaluate this distribution, one takes some particular values for $Y', T = y', t$ and computes the aggregates given by (1) and (2). We have assumed that behaviours are Boolean (they are either present or not), but Equation (4) can be extended to handle non-Boolean behaviours by also taking the conjunction with Behaviour'.*IUR.Behaviour'.value*.

There are two special behaviours that does not appear in the IU table at all: *DoNothing* and *Other*. *DoNothing* occurs either when no abilities are present for the current task, or when the state calls for doing nothing (a goal state or a state that does not appear in the IU table). The probability distribution over the *DoNothing* behaviour can be formulated using the same aggregations as for the other behaviours, negated:

$$P(DoNothing = true|Y, T) = \sum_{i \in I} [\neg row_abil_rel(i) \wedge row_rel(i)] \vee \prod_{i \in I} [\neg row_rel(i)] \vee goal \quad (5)$$

where $goal$ specifies the set of goal states.

Finally, to ensure non-zero probabilities, we add a small constant ρ to this distribution for any behaviour that is possible, and a second small weight, κ , on the behaviour if it was also present in the previous state. In our current implementation, $\rho = 0.01$ and $\kappa = 1$.

The behaviour *Other* handles regressions by the client in the task. An assumption made by the SNAP methodology [9] is that the only reason a client will not do an expected behaviour is because of a lack of the required abilities. Therefore, client behaviours are “atomic” by definition in the sense that the model cannot cope with multiple behaviours happening at once. However, the client can perform behaviours that cause a regression in the plan (e.g., when the client has finished washing her hands, she may take the soap again by mistake). In order to explain how our current implementation copes with regression, we use the example Bayesian network shown in Figure 6. This network shows generic nodes for the task states for two time steps: *previous state* and *state*, the ability, behaviour and observation. The problem of regression and non-atomic behaviours is mitigated by an additional artificial (non-existent in the IU analysis) behaviour *Other*. This behaviour does not require any abilities, and yields a state transition to all values of the state with a strong preference towards keeping all features with their existing values. With such a definition of behaviour *Other*, our system can adjust its belief state when the sensor readings provide information that contradicts the specified atomic behaviours, thereby alleviating the need for the designer to specify each and every possible regression.

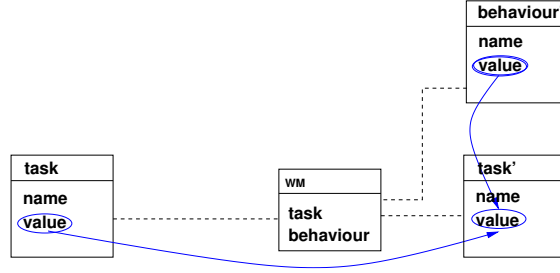


Figure 7: Relational schema and dependency structure for PRM for the client task dynamics, $P(t'|b', t)$. The state of the environment depends on its previous state and on the current behaviour of the client.

3.1.3. Task State Dynamics Model - T'

Figure 7 shows the dependency structure and relational schema for the PRM for the client task dynamics, $P(T'|B', T)$ (we leave off dependency on actions for clarity of presentation). In order to define the aggregation operators, we need to introduce a *link* class, WM, that represents the world model describing how client behaviours change the state of the world. A WM object has a single behaviour and a set of task variables (a state). We are seeking the CPT:

$$P(\text{Task'.Value} | \text{Pa}(\text{Task'.Value}))$$

and this is defined over an aggregate of the parents of Task'.Value , $\text{Task'.WM.Behaviour'}$ and Task'.WM.Task as follows:

$$P(T'|T, B') = \bigvee_{\substack{wm_j \in \\ \text{Task'.WM}}} \left(\begin{array}{l} wm_j.\text{Task'.Value} \wedge \\ wm_j.\text{Behaviour'.Value} \wedge \\ [\wedge wm_j.\text{Task'.Value}] \end{array} \right) \quad (6)$$

For each task variable value, t' , this function is a disjunction over all behaviours, b' that have t' as an effect, and each term in the disjunction is a conjunction over all task variables that are preconditions for b' , and further includes b' itself ($wm_j.\text{Behaviour'.Value}$) and the relevant value for t' ($wm_j.\text{Task'.Value}$ - only necessary if the task variable in question is non-Boolean).

3.1.4. Sensor Dynamics Model - K' and V'

For convenience, observations (sensors) are divided into states of sensors relevant to the task environment, K' , and states of sensors relevant to client behaviour, V' . We assume that each of the sensors depends on one (state or behaviour) variable only, and the sensor noises are represented explicitly in a table for each sensor and task variable combination.

3.1.5. Reward Model - R

The reward function is defined in a table that has a set of states and a reward value on each row. Action costs are specified relative to each ability the action is meant to help with, in the abilities tables.

3.2. Implementation Details

Figure 8 shows the structure of the entire database that stores our relational schemata, probabilistic dependencies, and information required to define conditional probability tables. Note that the database does not explicitly distinguish between primed and unprimed variables, i.e., it defines variables once, and then the interpretation of the model as a dynamic Bayesian network repeats every variable twice for two times slices. All tables that have their names starting with $t_iu_$ represent the IU table. The IU table view can be seen as a derived relation or table in the database: it is defined by an SQL join on the set of $t_iu_$ tables. As shown in Figure 5, the rows of the IU table do not define random variables (there are no probabilistic links to attributes of the IU table in Figure 5), rather they are used in order to define conditional probability tables for Behaviour'.Value . The task state attributes and their possible values are in table $t_env_variables_values$. The sensors are defined in $t_observations_values$ and are linked with the behaviour

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

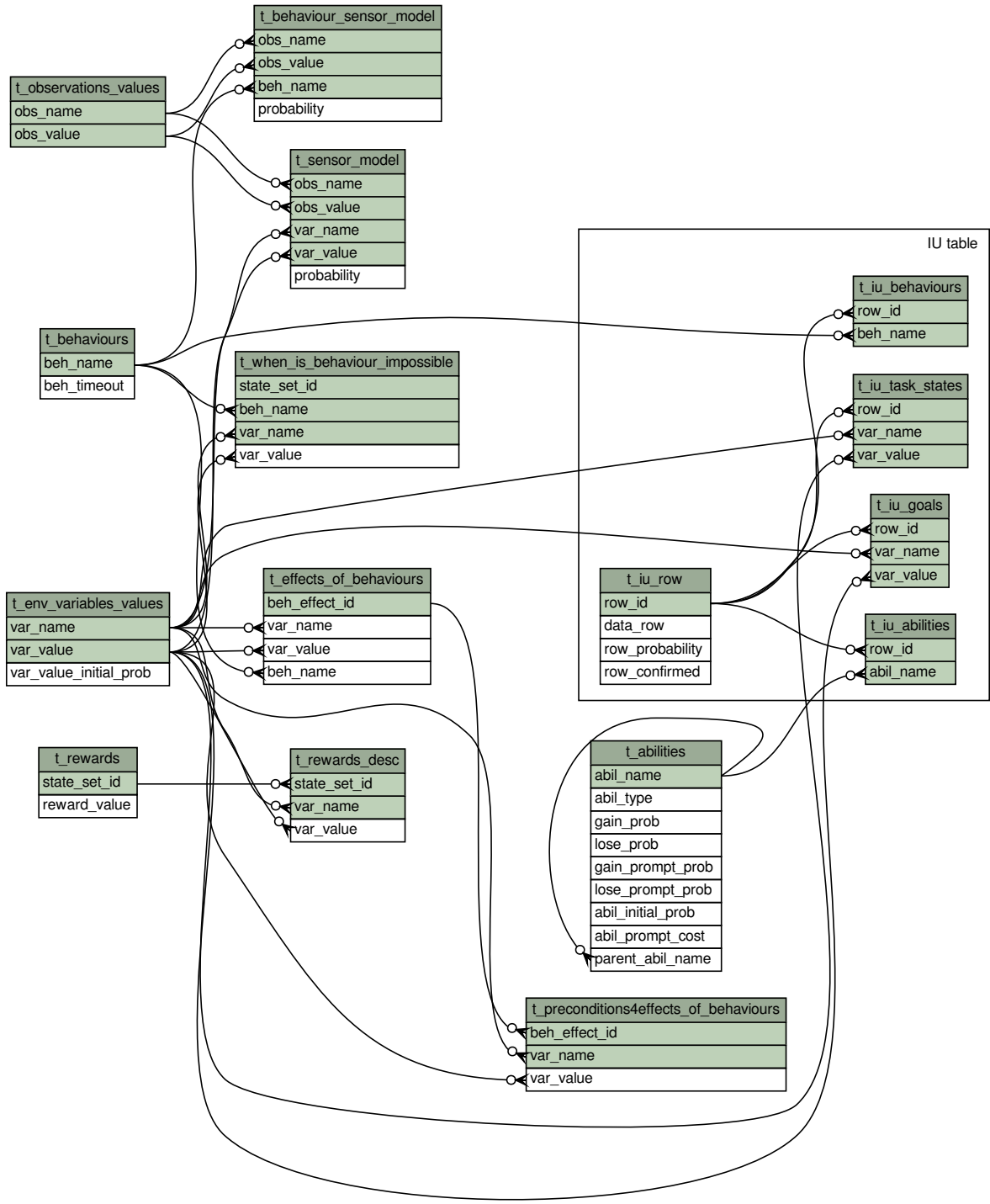


Figure 8: The diagram represents all tables which were implemented in the SQL database in order to store our SNAP model. Reference slots are indicated with lines connecting corresponding tables.

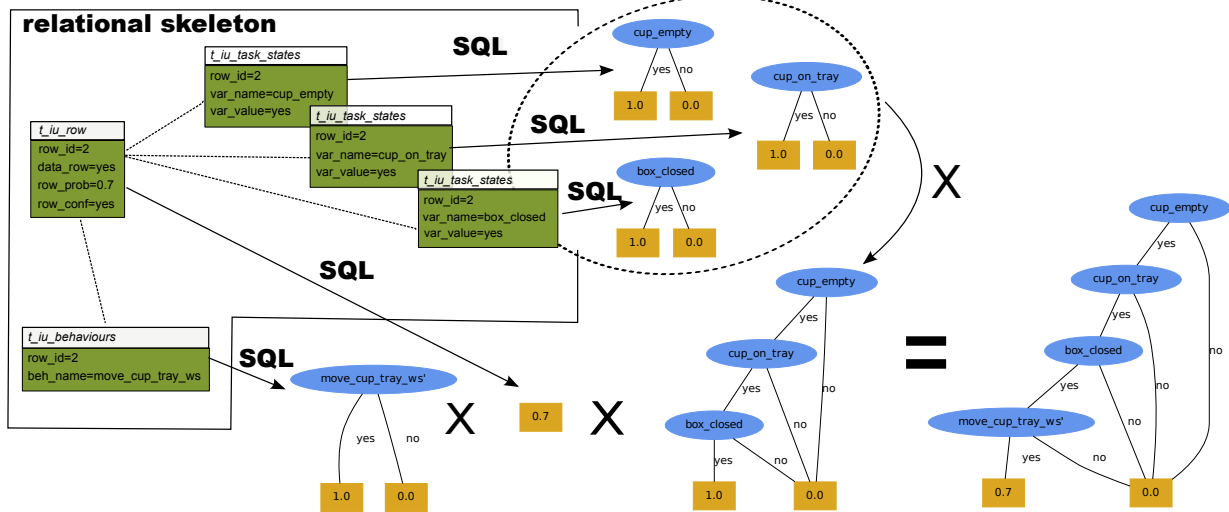


Figure 9: Subset of the relational skeleton for $P(b'|b, y, t')$ showing the generation of $\text{behaviour}(i, b') \wedge p(i) \wedge \text{row_rel}(i) = \text{row_rel}_b(i, b')$ for row $i = 2$ in Table 1. The algebraic decision diagrams (ADDs) representing the relations are multiplied and normalised to give the final CPT.

via `t_behaviour_sensor_model` or with the task via `t_sensor_model`. The last two tables are relationship classes that determine observations of the client and the task, and also define probabilities for sensor readings. The remaining tables that have `behaviour` in their name define dynamics of the client's behaviours. This includes effects and preconditions of behaviours, and also some additional constraints which specify when a behaviour is not possible. These behaviour related tables define what is shown in Figure 7 as the world model class. The preconditions and effects tables define the dynamics of behaviours that corresponds to what STRIPS operators [25] normally define in symbolic planning. Finally, rewards are defined in `t_rewards` and the associated table allows specifying sets of states which yield a particular value of the reward.

All the functions necessary to specify the POMDP are represented as algebraic decision diagrams (ADDs) in SPUDD notation [26]. These functions are computed with queries on the database. Each such query extracts some subset of ADDs from the relational skeleton. The ADDs are then combined using multiplication and addition to yield the final conditional probability tables (CPTs). Some relations are explicitly represented in the database whereas others need to be extracted using more complex SQL queries. For example, data for $\text{row_rel}(i)$, and $\text{row_abil_rel}(i)$ is read from the IU table in the database. The example subset of the relational skeleton for the IU analysis from Table 1, and diagrams for selected functions are in Figure 9. SQL queries extract compound algebraic decision diagrams from the relational skeleton, and the generator software multiplies those diagrams in order to obtain final functions, such as $P(b'|b, y, t')$. A schematic is shown in Figure 9, where the CPT for $\text{behaviour}'\text{move_cup_tray_ws}$ is gathered from the relevant tables. The original table schema in the PRM for the relations in Figure 9 can be seen in Figure 8.

3.3. Simulation and Validation

The database described in the last section is provided as an online service. When the entire model is specified in the database, the designer can generate a POMDP automatically, and download a text-file specification of the POMDP. We then provide software that solves the POMDP and allows for simulating it using a text-based input mechanism. Thus, the designer can experiment with how the model works in simulation, and in case of unexpected behaviour, the model can be adjusted and re-generated. This iterative process allows the designer to tune the reward function that would guarantee the desired behaviour of the system. The simulation process is a key step, as it allows the designer, who has knowledge of the task domain and target clients but not of POMDPs, to see if her task knowledge corresponds to the model she has created.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65

var_name	var_value
tap	on
tap	off
hands_w	dry
hands_w	wet
hands_c	soapy
hands_c	clean
hands_c	dirty
hw	yes
hw	no

t.env_variables_values

beh_name
other
nothing
dry_hands
turn_on_tap
turn_off_tap
rinse_hands
lather_hands
finish_handwashing

t_behaviours

abil_name
Rn_tap_off
Rn_tap_on
Af_alter_hands_c_to_clean
Af_alter_hands_c_to_soapy
Af_alter_hands_w_to_dry
Af_hw_yes

t_abilities

Figure 10: The content of tables that define task states, behaviours and abilities in our hand washing example.

3.3.1. Constraints and Database Engines

The advantage of the relational database is that it allows for easy implementation of the constraints required by the model. The simplest example are constraints on attribute values. For example, probabilities have to be in the range [0, 1], or literals should follow specific naming patterns (according to the requirements of the POMDP planner). These simple constraints are easily implemented in the definition of SQL tables. More complex constraints, which involve more than one attribute, are also required. For instance in the planner which we use, sensors and domain attributes are in the same name space, which means that their names have to be different. Furthermore, as we use directed graphical models, the acyclicity of our final DBN has to be guaranteed. Such things can be easily implemented using database triggers, and the designer will be prompted at the input time and informed about the constraint. The advantage of databases is that they allow for specifying and enforcing relations between different types of objects, for example, objects representing concepts in the environment and objects which serve as attributes of environment objects.

3.3.2. Hierarchical Control

The IU analysis breaks an ADL like making a cup of tea down into a number of sub-tasks, or sub-goals. For tea making, there are five sub-goals. This decomposition arises naturally according to the major elements of recall noted in the videos from which this IU analysis was made. The five sub-goals are partially ordered, and the partial ordering can be specified as a list of pre-requisites for each sub-goal giving those sub-goals that must be completed prior to the sub-goal in question. Since each sub-goal is implemented as a separate POMDP controller, a mechanism is required to provide hi-level control to switch between sub-goals. We have implemented two such control mechanisms. A deterministic controller is described in [9], and a probabilistic and hierarchical method in [27]. The hierarchical control is however beyond the scope of this paper, because here our interest is in how the POMDP specification of one sub-task can be rapidly specified.

4. Case Study with Encountered Pitfalls and Lessons Learned

In this section, we provide a case study that explains how our tool is used in order to design a prompting system to help a person with dementia to wash their hands. Our previous version of this system [3] used a hand-crafted POMDP model, and our goal here is to show the process of creating a similar, but not identical, model using the PRM models and database introduced in the last section. Normally, the IU analysis is completed in a spreadsheet [9], and here we follow a designer, SC, going through the IU analysis using our database. SC is an electrical engineer with no formal training in decision theory or Markov decision processes. As called for by the SNAP methodology [9], SC started this process by watching multiple videos of clients washing their hands with the help of human caregivers. He then identified the elements as described in the following subsections.

4.1. Task and Environment Features

In the first step, SC identified the task states, client abilities and behaviours, and saved them in corresponding tables. The exact content of these tables (where some attributes are omitted for clarity) after performing this step is in Figure 10. The designer is free to choose whatever names she pleases for the attributes.

beh_effect_id	var_name	var_value	beh_name
1	tap	on	turn_on_tap
2	hands_c	soapy	lather_hands
3	hands_c	clean	rinse_hands
4	hands_w	wet	rinse_hands
5	tap	off	turn_off_tap
6	hands_w	dry	dry_hands
7	hw	yes	finish_handwashing

t.effects_of_behaviours

beh_effect_id	var_name	var_value
1	tap	off
2	hands_c	dirty
3	hands_c	soapy
4	hands_c	soapy
3	tap	on
4	tap	on
5	hands_c	clean
5	tap	on
6	hands_w	wet
6	hands_c	clean
7	hands_c	clean
7	hands_w	dry
7	tap	off

t.preconditions4effects_of_behaviours

Figure 11: Dynamics of behaviours in the hand washing example.

IU	Task States	Abilities	Behaviours	Probability
8	hands_c=clean, hands_w=wet, tap=off	Af_alter_hands_w_to_dry	dry_hands	1
9	hands_c=clean, hands_w=wet	Af_alter_hands_w_to_dry	dry_hands	1

Figure 12: The example IU table where IU row 9 subsumes IU row 8. If the designer will not observe this, our system will do the check automatically.

4.2. Dynamics of Client Behaviours

Before specifying the IU table, SC specified details of behaviours of the client, where for every behaviour its preconditions and effects were specified in corresponding tables. The result of this step is in Figure 11. In these tables, every *beh_effect_id* corresponds to one effect/precondition pair of a given behaviour. For example, rows with *beh_effect_id*=7 could be translated to the following STRIPS-like notation:

```

Behaviour:      finish_handwashing
Precondition:   hands_c = clean AND hands_w = dry AND tap = off
Effect:         hw = yes

```

4.3. IU Table

As in the original paper-based IU analysis, after completing the above two steps, SC created the IU table that represents the core of the design. For that, a number of interaction units were determined (the granularity depends on the dementia level of the client, and also on other client specific factors and preferences), and every interaction unit corresponds to one row in the IU table. For every row, SC had to define the behaviour of the client, the task states that are relevant for that behaviour, and the abilities that are required for the behaviour to happen. It should be noted that the state relevance in the task state column usually makes stronger requirements about the behaviour than those specified in preconditions of the behaviour. This is because preconditions only say when the behaviour may be successful, whereas the IU table specifies when the behaviour will do something useful. Before we show the exact IU table that SC created in this case study, we discuss several pitfalls encountered in this step, lessons learned and our solutions.

4.3.1. State Subsumption in the IU Table

During the IU table specification, SC found that he needed to specify two interaction units with the same behaviour in the IU table that shared task states. This indicates an error because both interaction units correspond the same behaviour, and one of them is the subset of the other. Figure 12 shows an example where IU row 9 subsumes IU row 8. Our software indicates the presence of such dependencies and the designer needs to resolve them. In this case, SC added the state feature *tap=on* to row 9, resolving the issue.

IU	Task States	Abilities	Behaviours	Probability
1	hands_c=dirty	Af_alter_hands_c_to_soapy	lather_hands	1
2	tap=off	Rn_tap_off	turn_on_tap	1
3	hands_c=soapy, tap=on	Af_alter_hands_c_to_clean	rinse_hands	1
4	hands_c=clean, hands_w=wet	Af_alter_hands_w_to_dry	dry_hands	1
5	hands_c=clean, tap=on	Rn_tap_on	turn_off_tap	1
6	hands_c=clean, hands_w=dry, tap=off	Af_hw_yes	finish_handwashing	1

Figure 13: The initial IU table designed by the designer that contains behaviours that are have the same states relevant. For example, both *lather_hands* and *turn_on_tap* have the same relevant state *hands_c=dirty, tap=off* that is not captured by the existing IU table.

IU	Task States	Abilities	Behaviours	Probability
1	hands_c=dirty, tap=off	Rn_tap_off	turn_on_tap	0.65
2	hands_c=dirty, tap=off	Af_alter_hands_c_to_soapy	lather_hands	0.35
3	hands_c=soapy, tap=off	Rn_tap_off	turn_on_tap	1
4	hands_c=dirty, tap=on	Af_alter_hands_c_to_soapy	lather_hands	1
5	hands_c=soapy, tap=on	Af_alter_hands_c_to_clean	rinse_hands	1
6	hands_c=clean, hands_w=wet, tap=on	Rn_tap_on	turn_off_tap	0.2
7	hands_c=clean, hands_w=wet, tap=on	Af_alter_hands_w_to_dry	dry_hands	0.8
8	hands_c=clean, hands_w=wet, tap=off	Af_alter_hands_w_to_dry	dry_hands	1
9	hands_c=clean, hands_w=dry, tap=on	Rn_tap_on	turn_off_tap	1
10	hands_c=clean, hands_w=dry, tap=off	Af_hw_yes	finish_handwashing	1

Figure 14: The IU table from Figure 13 after automatic state expansion and removal of two generated rows that were undoing the progress towards the goal. Now, IU rows 1 and 2 are described by the same set of states that is relevant in two different behaviours and the likelihood of each is governed by the probability specified in column *Probability*.

4.3.2. Overlapping State Relevance

Sometimes there are multiple behaviours that can happen in a given state. For example, in Figure 13, behaviours *lather_hands* and *turn_tap_on* (from IU rows 1 and 2) have the same relevant state *hands_c=dirty, tap=off*. However, this particular state is not explicitly stated in the IU table, since state relevance is defined on partially instantiated sets of states. Although the designer correctly specifies two interaction units for the behaviours, she may not notice that these behaviours can happen in the same fully instantiated state. All states should be considered, states relevant for more than one behaviour identified, and states not relevant for any behaviour removed. For this reason, we introduced an automatic step in the process where the designer checks the subsumption of states and the overlapping state relevance. This second check automatically adds more rows to the IU table, so that all sets of states relevant for all of the alternative behaviours are specified, for cases with overlapping state relevance. After this step, there may be several interaction units in the IU table that have the same set of relevant task states, but different behaviours. When this process is applied to the IU table in Figure 13 the result is in Figure 14 (row 1 has become rows 2 and 4, while row 2 has become rows 1 and 3). The result of the above transformation means that the client may do one of multiple behaviours, and the designer has to specify probabilities for each (column *Probability* in the IU table). For example, the client may either turn on the tap or use the (pump) soap as a first step in handwashing (rows 1 and 2). Some clients may tend to start with turning the tap first, and the probability for that alternative would be higher as shown in Figure 14. Similarly, in rows 6 and 7, we see that the client can turn off the tap or dry her hands in either order, but is more likely to dry hands first. The need for specifying this probability is one of the reasons why the expansion of the IU table cannot happen in the background when the system is generating the POMDP from the database and interaction with the designer is required.

In this case study, SC was very confident with the features of the system explained above and he found their use rather intuitive and straightforward. Figure 14 shows the final IU table that SC created in the hand washing task discussed in this case study. Note that the task states are defined in terms of partially instantiated states (sets of states) and the full instantiation of all states was not required.

4.4. Rewards and Costs of Prompting

Once all the above steps are completed, the probabilistic dynamics of the core model are defined. The two missing things are rewards and sensors. The system allows the designer to specify the reward model explicitly in the database.

abil_name	gain_prob	lose_prob	gain_prompt_prob	lose_prompt_prob	abil_prompt_cost
Rn_tap_off	0.5	0.1	0.95	0.05	1
Rn_tap_on	0.5	0.1	0.95	0.05	1
Af_alter_hands_c_to_clean	0.4	0.2	0.95	0.05	0.5
Af_alter_hands_c_to_soapy	0.4	0.2	0.95	0.05	0.5
Af_alter_hands_w_to_dry	0.4	0.2	0.95	0.05	0.5
Af_hw_yes	0.4	0.2	0.95	0.05	0.5

Figure 15: The example content of the `t_abilities` table in one of our hand washing models. Ability `Af_hw_yes` models the client being able to see the affordance of finishing the hand washing task. The cost 0.5 for this ability should be noted.

Action Name	Value of the Action
donothing	350.349
prompt_Af_alter_hands_c_to_clean	349.795
prompt_Af_alter_hands_c_to_soapy	349.727
prompt_Af_alter_hands_w_to_dry	343.249
prompt_Af_hw_yes	350.456
prompt_Rn_tap_off	348.059
prompt_Rn_tap_on	349.295

Table 2: The value of actions that prompt for abilities defined in Figure 15 in the belief state that contains the marginal probability of variable hands washed, hw , $P(hw = yes) = 0.93$. Note that there is one prompt in this table for every ability from Figure 15. The action suggested by an optimal policy is in this case `prompt_Af_hw_yes`, i.e., an action with the highest value.

We found that the following advice was sufficient for the non-POMDP professional, SC, to specify the reward model:

- SC was informed that the reward for finishing the task should be 15 (this number is chosen arbitrarily as the largest reward, but any linear scaling of reward functions is possible).
- Additionally a designer can specify an intermediate reward for reaching an intermediate goal. This reward should be smaller than the reward for achieving the final goal state, or can be negative if the domain has some dangerous situations that should be avoided (this does not happen in the handwashing task).
- The third element of the reward model is the cost of prompting. This specifies how costly it is to prompt for each ability. The value of this cost may be client dependent, because some clients may not like being asked specific questions. In general, recognition prompts are more invasive with this regard, and designers are advised to use higher costs for recognition prompts.

SC was following the above guidance in this case study and he defined the following rewards: $R(hw=yes) = 15$ and $R(hand.c=clean) = 3$. The initial costs of abilities that SC provided were as shown in Figure 15. SC then noted that the resulting POMDP was issuing (in simulation) an unnecessary prompt for the ability `Af_hw_yes`, i.e., after the client finished washing hands with $hw=yes$ the system would issue one more prompt for `Af_hw_yes` that is required to make $hw=yes$. This ability states that the client can see the affordance of finishing the task. The specific POMDP value function that SC used in the simulation (with original costs of prompts as specified in Figure 15) is as shown in Table 2. The value of the prompt `Af_hw_yes` is slightly higher than the value of action `DoNothing`, but action `DoNothing` was the one that was expected by SC to happen in this particular situation. The exact full belief state is not printed here, but it had the marginal probability for $hw=yes$ being $P(hw=yes)=0.93$ that meant that the task has been completed with sufficiently high probability, and it was justified to require the `DoNothing` action from the system. When SC increased the cost of prompting for `Af_hw_yes` from 0.5 to 0.55, the system was correctly executing action `DoNothing` in such situations. This particular example shows how the designer can tune the reward function to suit his or her needs. The designer can simply increase the cost, thus pushing the system to wait for more evidence that would justify a more expensive prompt. When the designer is running simulations for a given POMDP, the values of all system actions are displayed in a similar way as in Table 2. This allows the designer to see the explanation for a specific behaviour and decide on the model change.

4.5. Sensors

The last step is to specify the technical details that concern the physical deployment of the system in the environment, and our assumption is that these steps will be performed by hardware engineers. They will specify what kinds

1
2
3 of sensors will be used, and they will provide the accuracy of those sensors that will be placed in tables that store the
4 sensor model for behaviours and task states.

5 In the following section, we complete the case study with a demonstration of the resulting system in practice,
6 along with simulated results for two other SNAP analyses.
7
8

9 **5. Demonstrative Examples**

10 We demonstrate the method on three assistance tasks: handwashing and toothbrushing with older adults with
11 dementia, and on a factory assembly task for persons with a developmental disability. We show that once our relational
12 system is designed (i.e. the database and the generator which reads the database and outputs the POMDP file), the
13 system is generic and allows the designer to deploy the system in different tasks simply by populating the database
14 for the new task. The IU analysis for handwashing and toothbrushing were performed by a designer, SC, whose
15 modelling work was reported in detail in our case study in Section 4. The analyses for the other two tasks were
16 performed by a biomedical engineer, with limited experience with POMDPs or planning in AI. As an example of the
17 power of our method, the factory task was coded in about six hours by the engineer. The factory task contained 6
18 different databases and associated POMDPs, each with about 5000 states, 24 observations, and 6 actions. This can
19 be compared to a manual coding of the system for handwashing (a smaller task), that took over 6 months of work
20 resulting in the system described in [28].
21

22 We clarify here that our experiments are not meant to demonstrate the final POMDP assistance systems working
23 with real clients. This is our eventual goal, but requires extensive additional work to set up clinical trials and recruit
24 clients that is usually only started once working systems have been completely tested in simulation and laboratory
25 environments. In this paper, we are focussed on giving designers the ability to specify the models, and so we test the
26 resulting models in simulations to check that they are consistent with the domains. We also test the hand washing
27 model in the real life deployment of the system with a human actor who served as a potential client of the system.
28 However, our SNAP analyses are performed using videos of real (potential) clients doing the actual tasks that we are
29 designing for.
30

31 *5.1. COACH and prompting*

32 Examples of automatic generation of task policies using IU analyses and a relational database were implemented
33 for the task of handwashing and toothbrushing. For these examples, people with mild to moderate dementia living
34 in a long term care facility were asked to wash their hands and brush their teeth in two separate trials. Videos were
35 recorded and then used in order to derive the model of their corresponding tasks.
36
37

38 *5.1.1. Handwashing*

39 The washroom used to capture video for IU analysis for this task had a sink, pump-style soap dispenser and towel.
40 Participants were led to the sink by a professional caregiver, and were encouraged to independently wash their own
41 hands. The IU analysis was performed on eight videos captured from a camera mounted beside the sink. Each video
42 was 2-4 minutes in length and involved one of six different clients with dementia (level not recorded). The task was
43 broken into five steps: 1) turn on tap; 2) get soap; 3) rinse hands; 4) turn off water; and 5) dry hands. Steps 1 and 2
44 can be completed in any order, followed by step 3. After completion of step 3, steps 4 and 5 can be completed in any
45 order. Figure 14 shows the IU table for the five steps and final virtual step (to provide an end condition), outlining
46 the overall goals of the task, environmental state, and the client's abilities and actions relevant to the task. The goal
47 `hands_c=clean` (hands condition = clean) implies that the hands have been physically cleaned and rinsed, while the
48 goal `hw=yes` (hands washed) indicates that the overall task is completed, which includes returning the environment to
49 the original state.
50

51 *5.1.2. Toothbrushing*

52 The videos used for the analysis captured participants in a washroom that had a sink, toothbrush, tube of toothpaste
53 and cup, as they tried to independently brush their own teeth. A formal caregiver was present to provide coaching and
54 assistance if required. Six videos were used with six participants. The IU analysis was completed based on videos
55 of several people and included multiple different methods of completing the task. The task was divided into 6 main
56
57
58

IU	Goals	Task States	Abilities	Behaviours
1	Final, wet brush	teeth_dirty, tap_off	Af: tap	Turn on tap
2	Final, wet brush	teeth_dirty, brush_on_surface	Rn: brush_on_surface	Take brush from surface
3	Final, wet brush	teeth_dirty, brush_in_cup	Rn: brush_in_cup	Take brush from cup
4	Final, wet brush	teeth_dirty, tap_on, brush_in_hand	Af: water	Wet brush

Table 3: IU analysis of step 1 (wet brush) in the toothbrushing task. Step 1 in this task is about turning the water on, taking the tooth brush and making the brush wet.

steps, each containing multiple sub-steps: 1) wet brush; 2) apply toothpaste; 3) brush teeth; 4) clean mouth; 5) clean brush; and 6) tidy up. Steps 1 and 2 can be completed in any order, followed by step 3. Steps 4 and 5 can also be completed in any order after step 3, and step 6 is the final step following the first 5 steps. Table 3 shows the IU table for step 1 (wet brush), displaying the task and subtask goals, environmental state, and the client’s abilities and actions relevant to the task.

5.1.3. Simulated Results on Hand Washing and Tooth Brushing

A policy was generated for the handwashing task and for each sub-step of the toothbrushing task entered. Simulations were run to test the handwashing and toothbrushing policies by entering observations that corresponded to the client always forgetting steps of the task throughout the simulation (i.e., doing nothing) but responding to prompting if provided. The simulations were run in a text-based interaction mode (no real sensors). Tables 4 shows a sample of the belief state of the POMDP, the system’s suggested action and the actual sensor states for several time steps during handwashing. Probabilities of the belief state are represented as the height of bars in corresponding columns of each time step. In the handwashing example, the client was prompted to get soap (t=1, *Af_alter_hands_c_to_soapy*). The client remained inactive (the same observations were received), so the system prompted the client to turn on the water (t=2, *Rn_tap_off*). The client turned on the water, so the system again prompted the client to get soap (t=3). After detecting that the client had taken soap and the tap was on (preconditions for the next step) the system prompted the client to rinse off the soap (t=4,5, *Af_alter_hands_c_to_clean*), which the client does at t=6. The client is then prompted to dry hands (t=6) and turn off the tap (t=7,8).

The toothbrushing simulation is shown in Table 5. At first, the system prompts the client to turn on the tap (t=1). The client does nothing, so the system tries to prompt the client to take the brush from the cup (in this case either turning the tap on or taking the toothbrush can happen first). The sensors indicate the brush was taken (t=3), so the system returns to prompting the client to turn on the tap.

5.1.4. Real-life Results on Hand Washing

The same POMDP and policy generated for the handwashing task simulations reported in Section 5.1.3 were used in real-world trials using the COACH system [28, 3]. The trials were run with SC impersonating an older adult with mild-to-moderate dementia. Figure 16 shows the client’s progression through the task at key frames in the trial. Each row of this figure shows the relevant belief state(s) in the center, and two images on either side showing how the computer vision system is tracking the hands (see [3] for details on these aspects). On the left, we see the overhead video used by the COACH system, overlaid with the (particle-filter based) trackers for hands and towel, and with the regions the hands are in highlighted. On the right, we see the color segmented image that is used by the tracker. For the trial, the policy is initialized when the client approached the sink (Figure 16-a), and the system prompts the client to get soap. The system correctly identifies that the client gets soap (Figure 16-b), and prompts to turn on the tap. The client turns on the tap (Figure 16-c) and the system prompts the user to rinse the soap off his hands. After scrubbing his hands for a short period of time, the system follows along as he rinses off the soap and turns off the tap (Figure 16-d) and the user is prompted to dry his hands. When he dries his hands (Figure 16-e) notifies the client that he has completed the task. After leaving the wash basin (Figure 16-f), the user is again notified that the task has been completed. This second notification is unnecessary, and can be fixed by reassigning a slightly higher cost for the final prompt.

5.2. Factory Assembly Task

In this example, workers with a variety of intellectual and developmental disabilities are required to complete an assembly task of a ‘Chocolate First Aid Kit’. This task is completed at a workstation that consists of five input slots,

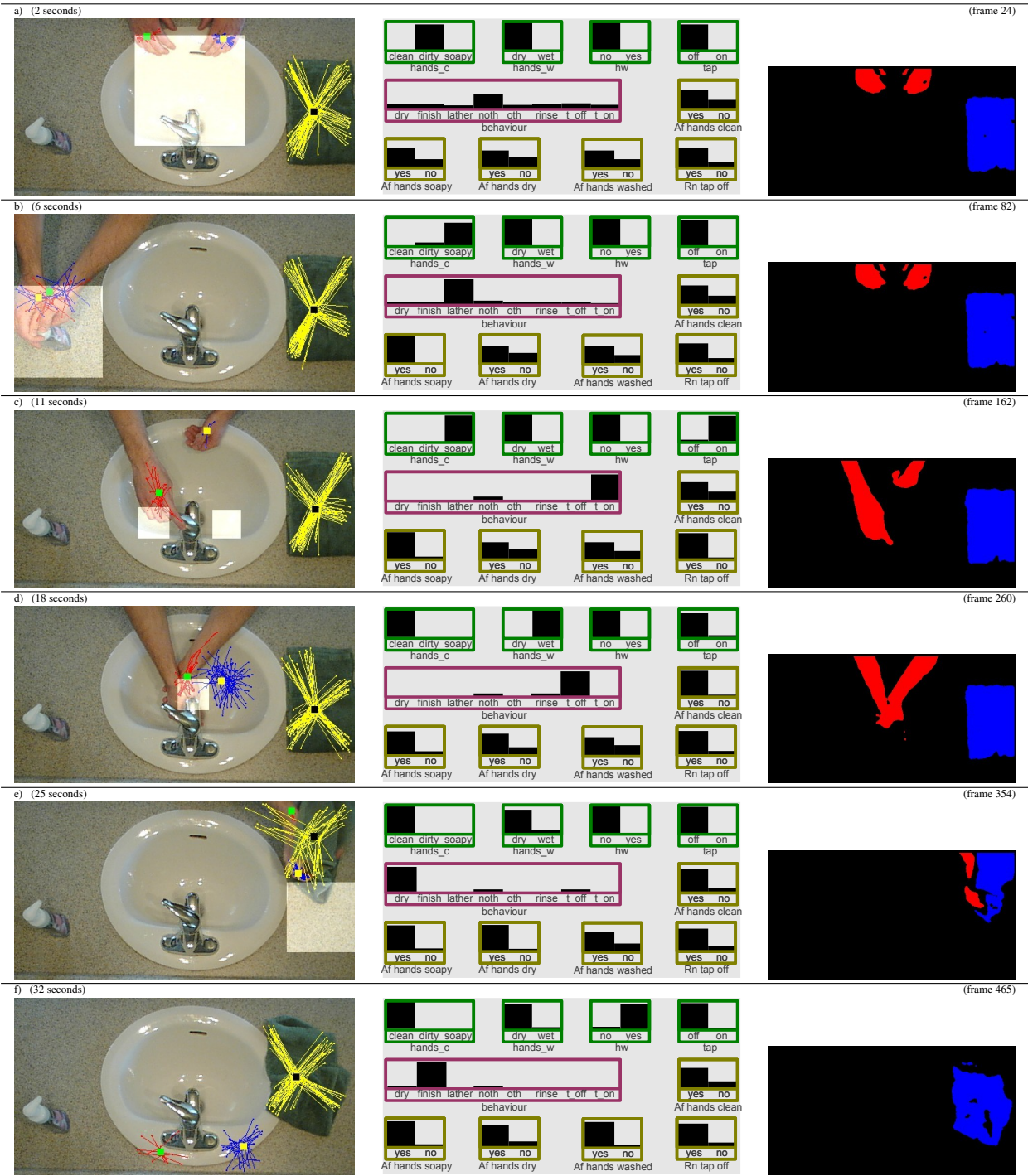


Figure 16: Key frames from the real-life hand washing experiment using the exact POMDP model that was designed by SC in the case study in Section 4 and simulated in Table 4, showing (left) the overhead video and computer vision hand/towel trackers (centre) marginal probabilities over task features, client abilities and behaviours (right) color segmentation image.

Step	Observations				Task				Behaviour				Ability				System Action (prompt for ability specified)											
	hands_c	hands_w	hw	tap	hands_c_clean	hands_c_dirty	hands_c_soapy	hands_w_dry	hands_w_wet	hw_no	hw_yes	tap_off	tap_on	dry_hands	finish_handwashing	lather_hands		nothing	other	rinse_hands	turn_off_tap	turn_on_tap	Af_alter_hands_c_to_clean	Af_alter_hands_c_to_soapy	Af_alter_hands_w_to_dry	Af_hw_yes	Rn_tap_off	Rn_tap_on
0	dirty	dry	no	off	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Af_alter_hands_c_to_soapy
1	dirty	dry	no	off	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Af_alter_hands_c_to_soapy
2	dirty	dry	no	off	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Rn_tap_off
3	dirty	dry	no	on	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Af_alter_hands_c_to_soapy
4	soapy	dry	no	on	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Af_alter_hands_c_to_clean
5	soapy	dry	no	on	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Af_alter_hands_c_to_clean
6	clean	wet	no	on	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Af_alter_hands_w_to_dry
7	clean	dry	no	on	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Rn_tap_on
8	clean	dry	no	on	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Rn_tap_on
9	clean	dry	no	off	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	Af_hw_yes
10	clean	dry	yes	off	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	donothing

Table 4: Example simulation in the handwashing task. All simulations are presented in the same way, where every row represents one time step. The sensor readings are shown in the observations column. The current belief estimation is reflected by the height of the bar in the columns corresponding to specific random variables (task, behaviour and ability). The last column shows the prompt suggested by the system.

an assembly area, and a completed area. The input slot contain all of the items necessary to perform kit assembly- specifically the main first aid kit container (white bin), and four different candy containers that need to be placed into specific locations within the kit container. The IU analysis was completed based on five videos of a specific adult worker (who has Down’s Syndrome) completing this assembly task with a human job coach present. Each video was 2-3 minutes in length. The worker was assessed with a moderate-to-mild cognitive impairment and was able to follow simple instructions from a job coach. The IU analysis broke this task into 6 required steps: 1) prepare white bin; 2) place in bin chocolate bottle 1; 3) place in bin chocolate box 1; 4) place in bin chocolate box 2; 5) place in bin chocolate bottle 2; and 6) finish output bin and place in completed area. Steps 2, 3, 4, and 5 can be completed in any order. Through a hierarchical task analysis [29] each of these steps were further broken down into sub-steps. Table 6 is an example IU analysis for step 2.

Policies were generated for each of the required assembly steps and were simulated by the designer for three different types of clients: mild, moderate, and severe cognitive impairment. Figure 17 is the output of sample timestamps for step 2 for a client with severe cognitive impairment. Again, probabilities of the belief state are represented as the height of bars in corresponding columns of each time step. In this specific example, the system is more active in its prompting based on the fact that the client is assumed to have diminished abilities with respect to the different aspects that needs to be completed. For example (t=1), the worker has deteriorating ability to recognize that the slot that holds the required chocolate bottle is empty. As such, the system correctly prompts the worker to recognize that the slot is empty and needs to be filled. In another example (t=5), the system recognizes that the worker has not placed the bottle in its correct location in the white bin, and provides a prompt for the person to recall that the bottle needs to be in that position in order to reach the final goal state. When the worker does not respond to this prompt, the system decides (t=6) to play a different, more detailed, prompt (a prompt related to the affordance ability).

6. Related Work

Below, we relate our work to the existing research on knowledge engineering for planning which is the main target of this paper and also to statistical relational learning which motivates our relational solution.

Time step, t	Observations	Task	Behaviour	Ability	System Action (prompt for ability specified)
	brush_wet tap brush_position	brush_wet brush_in_hand brush_in_cup brush_on_surface tap_on	other nothing alter_tap_to_on take_brush_from_cup wet_brush take_brush_from_surface	Rn_brush_in_cup Rn_brush_on_surface Af_tap Af_water	
0	dry off in_cup	---	---	■■■■■■■■	Af_tap
1	dry off in_cup	---	---	■■■■■■■■	Af_tap
2	dry off in_cup	---	---	■■■■■■■■	Rn_brush_cup
3	dry off in_hand	■	■	■■■■■■■■	Af_tap
4	dry on in_hand	■	■	■■■■■■■■	Af_water
5	dry on in_hand	■	■	■■■■■■■■	Af_water
6	wet on in_hand	■■■	■	■■■■■■■■	donothing

Table 5: Example simulation in the toothbrushing task. The goal in the shown sub-task is to turn on the tap, take the toothbrush from either the surface or the cup, and wet the brush.

IU	Goals	Task States	Abilities	Behaviours
1	Final assemble chocolate bottle 1	bottle1_in_other slot_orange_empty	Rn: slot_orange_empty	Fill slot orange
2	Final assemble chocolate bottle 1	bottle1_in_slot slot_orange_full	Rn: bottle1_in_slot	Move bottle1 from slot
3	Final assemble chocolate bottle 1	bottle1_in_hand slot_orange_full	Rl: bottle1_in_hand	Move bottle 1 to whitebin
4	Final assemble chocolate bottle 1	bottle1_in_whitebin slot_orange_full	Af: bottle1_correctposition	Alter bottle 1 to correct position

Table 6: IU analysis of step 2 in the factory assembly task. Step 2 of this task requires the client to fill in the orange slot with bottles when it is empty (the orange slot is the place from which bottles are taken and moved to the white bin). When the orange slot contains bottles, the client has to take one bottle, move it to the white bin, and then make sure that the bottle is in the correct position in the white bin.

6.1. Knowledge Engineering for Planning

We start this section with a reference to a more general approach of reinforcement learning (RL) which represents a broader class of planning problems where available domain knowledge is sufficient for only a partial formulation of the problem and the planning algorithm has to estimate the missing elements using simulation [30]. A recent survey of the existing tools and software for engineering RL problem specifications presented in [31] shows that engineering of RL domains is in most cases done either by re-implementation of required algorithms and domains/simulators or by partial re-use of the existing source code in the form of libraries or repositories, which means that engineering of RL domains is a direct implementation problem. There are no existing out of the box, domain independent environments where the specification of the RL problem would be reduced to the specification of the domain in a specific domain definition language.

Existing tools are far more advanced with this regard in the area of planning (both symbolic and MDP-based decision theoretic planning) where planners have been made publicly available and these planners can read the planning problem in a specific language (e.g., a variation of STRIPS is used in Graphplan [32] or SPUDD in the SPUDD planner [26]) directly and no coding of the planner is required from the user. Such planners are called domain independent planners [33], and it is sufficient for the designer of the planning domain to know the planner’s domain definition language and specify the domain in that language.

The process of defining the planning domain in the language of the planner can be still tedious or challenging for less experienced users and there has been past research which aims at providing tools which can help in the process of defining the planning problems [34, 35, 36]. The difference between these tools and our work is that they assume that the planning problem has been already identified by the designer and the goal of the tool is to help the

Time step, t	Observations		Task		Behaviour		Ability		System Action (prompt for ability specified)
	slot_orange_sensor	bottle1_position_sensor	bottle1_position_in_hand	bottle1_position_in_whitebin_pos1	other	nothing	R1_bottle1_position_in_whitebin_pos1	Rn_slot_orange_empty	
0	-	-	-	-	-	-	-	-	Rn bottle1 in slot
1	empty	other	-	-	-	-	-	-	Rn slot empty
2	empty	other	-	-	-	-	-	-	Rn slot empty
3	full	in_slot_orange	-	-	-	-	-	-	Rn bottle1 in slot
4	full	in_hand	■	-	-	-	-	-	R1 bottle1 in hand
5	full	in_whitebin	■	■	-	-	-	-	Af bottle1 to pos1
6	full	in_whitebin	■	■	-	-	-	-	R1 botte1 in bin
7	full	in_whitebin	■	■	-	-	-	-	Af bottle1 to pos1
8	full	in_whitebin_pos1	■	■	-	-	-	-	do nothing

Figure 17: Example simulation in the factory assembly task. The goal in the shown sub-task is to take the bottle, named bottle 1, from the orange slot and to place the bottle in the white bin in pos1.

designer in formalising the description using the specific formal language which is associated with the planner. Our work introduces a further separation between the designer and the planning domain specification by reducing the interaction between the designer and our tool to *the translation process* where the designer performs a psychological IU analysis of the task, enters the result into the database using a standard web-based interface, and then the planning problem definition of the prompting system is generated automatically from the content of the database. Actually, the designer does not have to be even fully aware that what he is doing in the database is the specification of the AI planning domain.

6.2. Probabilistic Relational Planning

First-order models allow for a compact representation of planning domains since the early days of symbolic planning where the STRIPS formalism is a good early example [25]. Various alternative and extended formulations were proposed such as PDDL [37] and its probabilistic version PPDDL [38]. Even though they are relational in their nature, it is not always easy to specify certain dependencies in the modelled domain. This was probably the reason why representations based on Dynamic Bayesian Networks (DBNs) were introduced [39]. Classical examples are SPUDD and SymbolicPerseus languages [26, 40]. They lack however first-order principles and things which are standard in STRIPS, such as parametrised actions, were not possible in existing DBN-based representations, until recently RDDDL combined ideas from models based on DBNs and variations of STRIPS [41]. This makes RDDDL an extended version of SPUDD/SymbolicPerseus [26, 40] with parametrised actions plus several other extensions. It is interesting to note that RDDDL applied the model of actions being DBN variables, which allows for concurrency but which exists also in prompting systems modelled using SPUDD, where client's actions - behaviours in our model - are variables in the DBN (see Figure 8 for details).

We now look how the above formalisms are related to our work. RDDDL becomes very convenient because conditional probability distributions and actions can be modelled in a compact way due to their parametrisation (like in STRIPS). Even though SPUDD/SymbolicPerseus language which we used does not have this property, we moved this task to the database and domain generator software which grounds actions on the fly and saves them to the SPUDD/SymbolicPerseus representation. The RDDDL based planner would need to ground actions at the planning level, and we do this at the intermediate representation level.

1
2
3
4 Initially, relational formalisms were used for a compact representation of the planning problem only, and planners
5 which were used for actual planning were grounded and did not exploit relational structure in the domain. In the
6 last decade, some solid—at least theoretically—work has emerged which aims at using relational representations
7 explicitly during planing in the so called lifted planning or first-order planning [42, 43, 44]. As long as relational
8 models are challenging to solve, their inherent properties can be useful in learning [45]. The work of this paper does
9 not aim at improving any specific POMDP planner, either lifted or grounded. Instead, the use of relational modelling
10 in our paper helps formulate the planning domain, and the fact that our methodology is based on a translation of a
11 psychological model in conjunction with relational modelling makes our tool accessible for designers who are not
12 experts in POMDP planning.

13 6.3. Statistical Relational Learning

14
15 The idea of Statistical Relational Learning (SRL) is to extend propositional probabilistic models (e.g., Bayesian
16 or decision networks) with the concept of objects, their attributes, and relations between objects. Two types of repre-
17 sentations are common in the SRL research: rule-based [46] and frame-based [21]. We base our work on the example
18 of the second type which is a Probabilistic Relational Model [11], because this model has an inherent connection with
19 a tabular representation of data which is found in relational databases.

20
21 The discussion of our model as a PRM established a natural connection with relational databases. It defines
22 objects, relations between objects and templates for probability distributions which can use aggregating operators in
23 order to deal with varying numbers of parameters. In our case, the parametrisation of objects and possible relations
24 between objects was also required and we achieved it by defining attributes of main domain objects, also as objects.
25 The resulting model encodes a template for specifying relational models, which when grounded, represent a specific
26 type of POMDPs model assistance tasks.

27
28 Similar requirements arise in statistical relational learning where analogous templates are necessary in order to
29 deal with ‘deep knowledge transfer’. This is the case in [47] where general transferable knowledge is extracted. The
30 second-order logic model defines the hypothesis space for machine learning algorithms in the same way as in our
31 frame-based model the template for assistance POMDPs is defined. If the logic- or rule-based SRL system does not
32 have to cope with ‘deep knowledge transfer’, first-order models are usually sufficient because they concentrate on
33 repeated structures which is satisfactory in order to provide the use of objects and relations between those objects
34 [48, 11, 49]. It is worth remembering that standard plate models also allow modelling repeated structures.

35
36 Lifted planning is still a challenge and currently most planners are grounded planners. A similar situation exists in
37 statistical relational learning where most actual reasoning or learning happens after grounding the model [50] though
38 recent work shows promising advancements in lifted reasoning [51, 52]. Regardless of specific architectures for
39 planning or statistical relational learning, i.e., grounded or lifted, relational models are very powerful in designing
40 probabilistic models.

41 7. Future Work

42
43 The SNAP methodology [9] was designed to enable non-technical users to specify POMDP prompting systems.
44 There is an inherent trade-off involved whereby the complexity of the model must be sacrificed for ease of specifica-
45 tion. The database we have presented in this paper implements only the most basic functionality, allowing a designer
46 to specify a prompting system for only the core set of assistive technologies (e.g. the handwashing system or other
47 simple tasks with atomic client behaviours, no exogenous events, and a restricted set of variable influences as shown
48 in Figure 1). In this section we discuss various extensions to this basic model that can be enabled by enlarging the
49 database and making specification of the model more complex.

50 7.1. Exogenous events

51
52 In the basic model, only the client can change the state of the world via his behaviours. However, in some cases the
53 state may be changed by exogenous events, such as the weather, clocks or other persons (e.g. caregivers). For example,
54 we are working on a system to help a person with dementia when they get lost while outside (wandering) [53, 54].
55 In this case, the task state may involve external elements such as weather, or proximity of a caregiver. The POMDP
56 models for this domain were obtained using the database system presented in this paper, and additional dependencies
57

1
2
3 were added easily, though the designer must specify some additional probabilities (e.g. the probability of arrival of a
4 caregiver, or the probability the weather will change).
5

6 7 *7.2. Sensing the cognitive state*

8 To apply our model to a dialogue system [55, 56], we need to allow for sensing the cognitive state of the client
9 (e.g. by asking a question). One of our projects is to add a dialogue component to an existing prompting system, and
10 we found that system actions were required to ask the client a question about her specific cognitive features. The
11 potential advantage of using our system for generating dialogue POMDPs should be explored more deeply in the
12 future, because it could allow for a straightforward integration of standard slot/filling dialogue systems with those
13 that require the model of the task. Such approaches are still challenging in research on dialogue systems [55] and our
14 methodology is addressing this missing link.
15

16 17 *7.3. Sensing the task state*

18 The situation mentioned in the previous point may also extend to the case when the system cannot sense a specific
19 task feature and would need to ask the client a question about the state of that feature. Appropriate system actions
20 would be required that would serve as sensors.
21

22 23 *7.4. Generalised Model of Assistance*

24 Our methodology and the existing software do not have to be limited to particular types of assistive systems that
25 we presented so far in the paper. In order to show this fact, we briefly discuss here a generalised model of human
26 assistance that could be used in a broad range of applications where the cognitive model of the human being is required
27 or useful. The first extension to the original model is that a more general cognitive state of the client is required that
28 has a broader meaning than the *abilities* used in this paper. Another extension is that, in some applications such as
29 tutoring systems [57], it is useful to model the cognitive behaviours of the client, and for this reason, the behaviour
30 should be applicable for both physical (i.e., belonging to the environment or real world) and cognitive behaviours of
31 the client. For example, in a tutoring system, the cognitive behaviour of the client reaching the understanding of a
32 new concept would trigger a change in the cognitive state of the client. The dependency of the cognitive state on the
33 behaviour does not require changes to our system, because the cognitive state that does require this feature could be
34 placed in the same table as the standard task state.
35

36 37 **8. Conclusions**

38 POMDP models have proven to be powerful for modelling intelligent human assistance [3]. Unfortunately,
39 POMDPs, being propositional models, usually require a very labour intensive, manual set-up procedure. Recent
40 work has shown how this can be simplified by using a methodology from the psychology literature called interaction
41 unit (IU) analysis [9]. In this paper, we show how the introduction of a probabilistic relational model and a database
42 encoding can greatly simplify and standardise this process. We derived a methodology for specifying POMDPs for
43 intelligent human assistance which is motivated by relational modelling in frame-based SRL methods [21]. The core
44 of our approach is the relational database which the designer populates in order to prepare the deployment of the
45 system for a specific task.
46

47 Traditionally, the specification of POMDPs was performed by trained POMDP experts who would encode the
48 problem in a specific notation that the planner can understand. The existing research investigated the problem of
49 how this specification can be done by non-POMDP experts, where the specialised tool helps designing the POMDP.
50 The key feature of such systems is that the designer still explicitly defines the planning problem. In this paper, we
51 implement another way of defining POMDPs through a translation of a psychological model. The designer is doing the
52 psychological task analysis of the task, whereas the system takes the data provided by the designer and automatically
53 translates it into the valid POMDP in the background. This results in a new paradigm that frees the designer from the
54 burden of knowing the POMDP technology. Our design is facilitated by the correspondence between the psychological
55 Interaction Unit (IU) analysis and POMDPs for assistive systems [9], and by statistical relational modelling in artificial
56 intelligence.
57

1
2
3
4 In our system, content provided by the designer of a particular implementation encodes the goals, action pre-
5 conditions, environment states, cognitive model, client and system actions, as well as relevant sensor models, and
6 automatically generates a POMDP model of the assistance task being modelled. The strength of the database is that it
7 also allows constraints to be specified, such that we can verify the POMDP model is valid for the task. To the best of
8 our knowledge, this is the first time the POMDP planning problem is formalised using a relational database.

9 We demonstrate the method on three assistance tasks: handwashing and toothbrushing for elderly persons with
10 dementia, and on a factory assembly task for persons with a cognitive disability. This demonstration shows that
11 the system, once designed using the relational approach, can be instantiated to create a POMDP controller for an
12 intelligent human assistance task. The use of the relational database makes the process of specifying POMDP planning
13 tasks straightforward and accessible to non-expert computer users. We have applied this method to other tasks as well,
14 including location assistance and dialogue management for scheduling applications.

15 We currently release our POMDP representation and solution software as an open-source distribution. Upon
16 publication of this paper, we will further release our POMDP generation software. Further, our system is available
17 online for testing by interested users using a secure web-based access system. This allows a user to create an account
18 on our system, to provide the data for his/her task, and then to generate a valid POMDP model for his/her task without
19 any software downloads¹.

20 21 22 **9. Acknowledgements**

23 This research was sponsored by American Alzheimer’s Association, the Toronto Rehabilitation Institute (TRI),
24 and by NIDRR as part of the RERC-ACT at ATP Partners, University of Colorado-Denver. The first author was
25 supported by a fellowship from the Ontario Ministry of Research and Innovation.

26 27 28 **References**

- 29
30 [1] T. Gill, B. Kurland, The burden and patterns of disability in activities of daily living among community-living older persons, *Journal of*
31 *Gerontology Series A: Biological Sciences and Medical Sciences* 58A (1) (2003) M70–M75.
32 [2] A. Burns, P. Rabins, Carer burden in dementia, *International Journal of Geriatric Psychiatry* 15 (1) (2000) 9–13.
33 [3] J. Hoey, P. Poupart, A. von Bertoldi, T. Craig, C. Boutilier, A. Mihailidis, Automated handwashing assistance for persons with dementia using
34 video and a partially observable Markov decision process, *Computer Vision and Image Understanding* 114 (5) (2010) 503–519.
35 [4] J. Hoey, P. Poupart, C. Boutilier, A. Mihailidis, POMDP models for assistive technology, in: E. Sucar, E. Morales, J. Hoey (Eds.), *Decision*
36 *Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, IGI Global, 2011.
37 [5] L. Chen, C. D. Nugent, M. Mulvenna, D. Finlay, X. Hong, M. Poland, A logical framework for behaviour reasoning and assistance in a smart
38 home, *International Journal of Assistive Robotics and Mechatronics* 9 (4) (2008) 20–34.
39 [6] W. Pentney, M. Philipose, J. Bilmes, Structure learning on large scale common sense statistical models of human state, in: *Proc. AAAI*,
40 Chicago, 2008.
41 [7] F. Mastrogiovanni, A. Sgorbissa, R. Zaccaria, An integrated approach to context specification and recognition in smart homes, in: *Smart*
42 *Homes and Health Telematics*, Springer, 2008, pp. 26–33.
43 [8] D. Salber, A. Dey, G. Abowd, The context toolkit: Aiding the development of context-enabled applications, in: *Proc. of the Conference on*
44 *Human Factors in Computing Systems (CHI)*, Pittsburgh, 1999, pp. 434–441.
45 [9] J. Hoey, T. Plötz, D. Jackson, A. Monk, C. Pham, P. Olivier, Rapid specification and automated generation of prompting systems to assist
46 people with dementia, *Pervasive and Mobile Computing* 7 (3) (2011) 299–318, doi:10.1016/j.pmcj.2010.11.007.
47 [10] H. Ryu, A. F. Monk, Interaction unit analysis: A new interaction design framework, *Human-Computer Interaction* 24 (4) (2009) 367–407.
48 [11] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, B. Taskar, *An Introduction to Statistical Relational Learning*, MIT Press, 2007, Ch. 5: Probabilistic
49 Relational Models.
50 [12] K. J. Åström, Optimal control of Markov decision processes with incomplete state estimation, *Journal of Mathematical Analysis and Appli-*
51 *cations* 10 (1965) 174–205.
52 [13] P. Poupart, An introduction to fully and partially observable markov decision processes, in: E. M. Enrique Sucar, J. Hoey (Eds.), *Decision*
53 *Theory Models for Applications in Artificial Intelligence: Concepts and Solutions*, IGI Global, 2011, pp. 33–62.
54 [14] J. Hoey, P. Poupart, C. Boutilier, A. Mihailidis, POMDP models for assistive technology, in: *Proc. AAAI Fall Symposium on Caring*
55 *Machines: AI in Eldercare*, 2005.
56 [15] A. Mihailidis, J. Boger, M. Candido, J. Hoey, The COACH prompting system to assist older adults with dementia through handwashing: An
57 efficacy study, *BMC Geriatrics* 8 (28).
58 [16] B. Kirwan, L. Ainsworth, *The task analysis guide*, Taylor and Francis, London, 1992.

59
60
61
62
63
64
65

¹See <http://www.cs.uwaterloo.ca/~jhoey/research/snap/> for an overview of the project, including a video demonstration and a link to the online system

- 1
2
3
4 [17] D. Duke, P. Barnard, D. Duce, J. May, Syndetic modelling, *Human-Computer Interaction* 13 (4) (1998) 337.
- 5 [18] J. P. Wherton, A. F. Monk, Problems people with dementia have with kitchen tasks: the challenge for pervasive computing, *Interacting with Computers* 22 (4) (2009) 253–266.
- 6 [19] L. McCluskey, Knowledge engineering for planning roadmap, unpublished Project Report (2000).
- 7 [20] S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2010.
- 8 [21] L. Getoor, B. Taskar (Eds.), *Statistical Relational Learning*, MIT Press, 2007.
- 9 [22] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.*, Morgan Kaufmann, San Mateo, CA, 1988.
- 10 [23] D. Heckerman, J. S. Breese, A new look at causal independence, in: *In Proc. of the Tenth Conference on Uncertainty in Artificial Ingelligence*, 1994, pp. 286–292.
- 11 [24] F. J. Díez, M. J. Druzdźel, Canonical probabilistic models for knowledge engineering, Technical Report CISIAD-06-01, UNED, Madrid, Spain (2006).
- 12 [25] R. Fikes, N. Nilsson, STRIPS: a new approach to the application of theorem proving to problem solving, *Artificial Intelligence* 2 (1971) 189–208.
- 13 [26] J. Hoey, R. St-Aubin, A. Hu, C. Boutilier, SPUDD: Stochastic planning using decision diagrams, in: *Proceedings of Uncertainty in Artificial Intelligence*, Stockholm, 1999, pp. 279–288.
- 14 [27] J. Hoey, M. Grześ, Distributed control of situated assistance in large domains with many tasks, in: *Proc. of ICAPS*, Freiberg, Germany, 2011.
- 15 [28] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, A. Mihailidis, A planning system based on Markov decision processes to guide people with dementia through activities of daily living, *IEEE Transactions on Information Technology in Biomedicine* 10 (2) (2006) 323–333.
- 16 [29] R. Stammers, A. Sheppard, *Evaluation of Human Work*, 2nd Edition, Taylor & Francis, 1991, Ch. Chapter 6: Task Analysis.
- 17 [30] D. P. Bertsekas, J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.
- 18 [31] T. Kovacs, R. Egginton, On the analysis and design of software for reinforcement learning, with a survey of existing systems, *Machine Learning* 84 (1–2) (2011) 7–49.
- 19 [32] A. L. Blum, M. L. Furst, Fast planning through planning graph analysis, *Artificial Intelligence* 90 (1997) 281–300.
- 20 [33] M. Ghallab, D. Nau, P. Traverso, *Automated Planning, Theory and Practice*, Elsevier, Morgan Kaufmann Publishers, 2004.
- 21 [34] S. Edelkamp, T. Mehler, Knowledge acquisition and knowledge engineering in the modplan workbench, in: *Proc. of the First International Competition on Knowledge Engineering for AI Planning*, 2005.
- 22 [35] R. Simpson, D. E. Kitchin, T. McCluskey, Planning domain definition using GIPO, *Knowledge Engineering Review* 22 (2) (2007) 117–134.
- 23 [36] T. Vacquero, F. Tonidanel, J. Silva, The itSIMPLE tool for modelling planning domains, in: *Proc. of the First International Competition on Knowledge Engineering for AI Planning*, 2005.
- 24 [37] M. Helmert, PDDL resources, <http://ipc.informatik.uni-freiburg.de/PDDLResources> (2009).
- 25 [38] H. Younes, M. Littman, PPDDL: The probabilistic planning domain definition language, <http://www.cs.cmu.edu/~lorens/papers/ppddl.pdf> (2004).
- 26 [39] T. Dean, K. Kanazawa, A model for reasoning about persistence and causation., *Computational Intelligence* 5(3) (1989) 142–150.
- 27 [40] P. Poupart, Exploiting structure to efficiently solve large scale partially observable Markov decision processes, Ph.D. thesis, University of Toronto, Toronto, Canada (2004).
- 28 [41] S. Sanner, Relational dynamic influence diagram language (RDDL): Language description, http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/RDDL.pdf (2010).
- 29 [42] C. Boutilier, R. Reiter, B. Price, Symbolic dynamic programming for first-order MDPs, in: *IJCAI*, 2001, pp. 690–700.
- 30 [43] K. Kersting, M. V. Otterlo, L. D. Raedt, Bellman goes relational, in: *Proc. of ICML*, 2004.
- 31 [44] S. Sanner, First-order decision-theoretic planning in structured relational environments., Ph.D. thesis, University of Toronto (2008).
- 32 [45] T. Lang, M. Toussaint, K. Kersting, Exploration in relational worlds, in: *Proc. of ECML/PKDD*, 2010, pp. 178–194.
- 33 [46] T. Sato, A statistical learning method for logic programs with distribution semantics, in: *Proc. of ICLP*, MIT Press, 1995, pp. 715–729.
- 34 [47] J. Davis, P. Domingos, Deep transfer via second-order Markov logic, in: *Proc. of ICML*, 2008.
- 35 [48] K. B. Laskey, MEBN: A language for first-order Bayesian knowledge bases., *Artificial Intelligence* 172 (2008) 140–178.
- 36 [49] D. Koller, A. Pfeffer, *Object-oriented Bayesian networks*, in: *Proc. of UAI*, 1997.
- 37 [50] P. Domingos, D. Lowd, *Markov Logic: An Interface Layer for Artificial Intelligence*, Morgan & Claypool, 2009.
- 38 [51] G. V. den Broeck, N. Taghipour, W. Meert, J. Davis, L. D. Raedt, Lifted probabilistic inference by first-order knowledge compilation, in: *Proc. of IJCAI*, 2011.
- 39 [52] V. Gogate, P. Domingos, Probabilistic theorem proving, in: *Proc. of UAI*, 2011.
- 40 [53] J. Hoey, X. Yang, E. Quintana, J. Favela, Lacasa: location and context aware safety assistant, in: *Proc. of UbiHealth*, 2012.
- 41 [54] J. Hoey, X. Yang, J. Favela, Decision theoretic, context aware safety assistance for persons who wander, in: *Proc. of Pervasive Workshop on Healthcare*, 2012.
- 42 [55] M. Frampton, O. Lemon, Recent research advances in reinforcement learning in spoken dialogue systems, *The Knowledge Engineering Review* 24 (04) (2009) 375–408.
- 43 [56] J. D. Williams, Decision theory models for applications in artificial intelligence: Concepts and solutions, in: E. M. Enrique Sucar, J. Hoey (Eds.), *Decision Theory Models for Applications in Artificial Intelligence*, IGI Global, 2011.
- 44 [57] R. C. Murray, K. VanLehn, *DT Tutor: A decision-theoretic, dynamic approach for optimal selection of tutorial actions*, in: C. F. G. Gauthier, K. VanLehn (Eds.), *Fifth international conference on Intelligent tutoring systems*, Springer, New York, 2000, pp. 153–162.