A Logic-Based Methodology for the Formalization of Shikake Principles and Examples

Daniela Inclezan
Texas Tech University
Lubbock, Texas 79410
daniela.inclezan@gmail.com

Abstract

Shikake is a design approach that proposes solving problems by inducing spontaneous behavior, rather than by relying on the use of extensive resources or expertise. This paper contributes to the study of *Shikake* principles and examples by describing a methodology for their formalization in the declarative, logic-based language of Answer Set Prolog (ASP). Modeling qualitative theories and principles such as *Shikake* in the precise language of ASP can play a significant role in indicating possible areas for their future refinement and improvement, as shown here. Our formalization is used in creating a system, SHASP, that can automatically determine if a design is a *Shikake* or not, as illustrated by two examples and one counterexample.

Introduction

Shikake is a design approach that proposes solving problems by inducing spontaneous behavior, rather than by relying on the use of extensive resources or expertise. Its name comes from the Japanese word "shikake," which means "physical and/or psychological mechanisms that trigger implicit or explicit behavior-change" (Matsumura 2012). The main advantages of the *Shikake* design method are its "low expertise, low cost, wide range of target users, and long term continuous behavior changes" (Matsumura 2012). Here are a couple of examples from (Matsumura 2012).

Example 1 *[Tennouji Zoo] A cylinder is located beside a pathway at Tennouji Zoo. The cylinder attracts people's attention, makes them look inside, provides them with discoveries, and consequently gives them pleasure.*

Example 2 *[Back Street] A tiny shrine gate symbol is placed on building walls on a back street. The tiny shrine gate gives people the holy impression around the place, makes people hesitate to do bad behavior, and eventually reduces surrounding garbages.*

This paper contributes to the study of *Shikake* principles and examples by describing a methodology for their formalization in the declarative, logic-based language of Answer Set Prolog (Gelfond and Lifschitz 1988; 1991). Answer Set Prolog (ASP) provides means for an elegant and accurate representation of commonsense knowledge and reasoning mechanisms. Recent papers have demonstrated that ASP is a useful tool for the refinement of qualitative theories and principles that are expressed in natural language (Balduccini and Girotto 2010; Inclezan 2012). Modeling such theories and principles in a precise language like ASP has the potential of pointing out the ambiguous and vague aspects of their English descriptions, thus indicating directions for theory/ principle revision. Modeling *Shikake* in ASP can contribute to the clarification and further extension of its principles.

ASP has several other advantages relevant to the formalization of *Shikake* principles and examples. There are established ASP methodologies for the concise and clear representation of *default statements* (i.e., statements of the type *Normally birds fly*) and their exceptions (e.g., *Penguins do not fly*) (Baral and Gelfond 1994). This is especially relevant for the formalization of psychological and sociological knowledge about the default behavior of different categories of people, which is one of the important aspects taken into consideration by *Shikake* design. Some examples of possibly relevant default statements are: *Normally children are curious*, or *Generally, people mimic the behavior of other people around them*. Another advantage of ASP is that it has well-studied methods for modeling evolving domains in which change is caused by action, in particular the effects of actions, action preconditions, and triggered actions (Gelfond and Lifschitz 1998; Baral 2003; Balduccini and Gelfond 2003). These methods can be used to encode cause and effect relationships, and create dynamic models of the interaction of different target users with *Shikake* designs. ASP can also represent ontologies and reason with uncertainty. This can be useful, for instance, in creating a taxonomy of "things" (e.g., *hard things* and *things with holes/ perforations* as subclasses of *things*; *wooden things* as a subclass of *hard things*; etc.) and their possible uses. Such knowledge would be especially relevant in creating knowledge-based systems for automated *Shikake* design.

The resulting formalization of *Shikake* principles, which benefits from the mentioned advantages of ASP, was successfully used to create a system called SHASP that mimicks part of the capabilities of a designer of *Shikake*. In the current stage, the intelligent agent is able to determine whether different scenarios represent *Shikake* examples or not. In the

future, we expect to endow SHASP with capabilities for inventing its own *Shikakes*.

In what follows, we give an introduction to ASP, describe our encoding of background knowledge relevant to our task, and present our formalization of *Shikake* principles. We describe an ASP encoding of scenarios and test our formalization on these scenarios. We briefly describe system SHASP, and end with conclusions and directions for future work.

Answer Set Prolog

Answer Set Prolog (ASP) (Gelfond and Lifschitz 1988; 1991) is a declarative, logic-based, knowledge representation language. A logic program of ASP can be seen as a specification for the set(s) of beliefs held by a rational agent. The vocabulary used to describe an agent's beliefs is represented by a *signature* — a collection of constant, function, predicate, and variable symbols. Terms and atoms over this signature are defined as in predicate logic. A literal is an atom a or its negation $\neg a$, where the symbol "$\neg$" denotes strong negation. A rule over a signature Σ is a statement of the form:

$$l_1 \ or \ \ldots \ or \ l_i \ \leftarrow \ l_{i+1}, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

where l_1, ..., l_n are literals of Σ, the symbol "*or*" denotes epistemic disjunction, and the symbol "not" denotes default negation. For instance, the rule

$$a \ or \ \neg b \ \leftarrow \ c, \neg d, \text{not } e$$

intuitively says: *Believe that a is true or that b is false if you believe that c is true, and believe that d is false, and have no reason to believe that e is true."*

A *program* of ASP is a pair $\langle \Sigma, \Pi \rangle$, where Σ is a signature and Π is a collection of rules over Σ.

For the formal semantics of programs of ASP we refer the reader to (Gelfond and Lifschitz 1991). For brevity, we only include here the informal semantics. An *answer set* A of a logic program $\langle \Sigma, \Pi \rangle$ is a consistent set of literals of Σ that corresponds to a set of beliefs held by the rational agent, such that A satisfies the rules of Π and the rationality principle: *Believe nothing you are not forced to believe.*

One main advantage of ASP is that different reasoning tasks can be reduced to computing answer sets of a logic program, which can be done using general purpose ASP solvers, for example CLASP (Gebser et al. 2007). The grounder GRINGO of CLASP is capable of understanding an additional type of rule, *choice rules*. An example of a choice rule is:

$$1 \ \{p(X) : q(X)\} \ 1 \ \leftarrow \ cond$$

which says: *If condition $cond$ is satisfied by an answer set A, then A must contain exactly one atom of the form $p(X)$, and if $p(t) \in A$, then $q(t)$ must also belong to A.*

Modeling the Relevant Background Knowledge

The *Shikake* methodology assumes that designers possess some background knowledge concerning things in the world and their possible uses, the dynamics of the world, as well as basic psychological and/or sociological knowledge about the default behavior of people. In this section, we discuss the application of well known ASP methodologies for knowledge representation, with the goal of creating a relevant background knowledge base for our purpose, which will be captured by an ASP program denoted by $\mathcal{K}$.

Ontology of Things and Their Possible Uses

We assume that the designer's background knowledge contains information about classes of things organized into a DAG-like inheritance hierarchy. We consider a possible ontology of this type containing, for instance, classes $entities$, $people$, $children$, $adults$, $hard\_things$, $wooden\_things$, $things\_with\_perforations$, $drawings$, $locations$, $public\_locations$, etc., and commonsense subclass relationships between them. Let us call this hierarchy H. We capture it in ASP by using a common approach (Baral 2003). The signature of our knowledge base, $\Sigma(\mathcal{K})$, contains: the names of all classes in the ontology; a sort called $class$; the relation $is\_subclass(C_1, C_2)$ (C_1 is a subclass of C_2); and its transitive closure $subclass(C_1, C_2)$. Program $\mathcal{K}$ contains rules defining the relation $subclass$ and facts of the type:

$$is\_subclass(people, entities).$$

It is normally the case that the description of a hierarchy is accompanied by the specification of one of its instances. An *instance of a hierarchy* is an interpretation of its classes, given a universe, such that it respects the $subclass$ relation, and the interpretation of the root class is equal to the universe. To be able to represent instances of our hierarchy, we extend $\Sigma(\mathcal{K})$ by the sort $object$, the relation $is\_a(X, C)$ (object X is a C) denoting links between objects and classes, and $inst(X, C)$ (X is an instance of C) denoting the closure of is_a. We add to $\mathcal{K}$ rules for $inst$. We assume that each *Shikake* scenario will provide a specific instance of the hierarchy by defining the extension of the relation is_a.

For the purpose of encoding *Shikake* principles, it is important to consider the possible uses of things, and thus it is relevant to express default statements of the type: *Typically, instances of class C can be used as R.* For that, we import the concept of a *role* from (Fan et al. 2001), defined as a "thing that is, but only in the context of things that happen" (i.e., only in the context of an event). We expand $\Sigma(\mathcal{K})$ by the sort $role$ and add to $\mathcal{K}$ facts of the type:

$$role(focusing\_tool). \qquad role(pounding\_tool).$$
$$role(fire\_starting\_tool). \quad role(symbol).$$

Then, we use a known methodology for representing defaults (Baral and Gelfond 1994). Specifically, we add to $\Sigma(\mathcal{K})$ the relations $capability(X, R)$ (X has the capability of being used according to role R) and $ab(X, R)$ (X is abnormal with respect to role R). $\mathcal{K}$ will now contain rules similar to:

$$capability(X, focusing\_tool) \leftarrow$$
$$\quad inst(X, things\_with\_perforations),$$
$$\quad \text{not } ab(X, focusing\_tool).$$
$$capability(X, pounding\_tool) \leftarrow$$
$$\quad inst(X, hard\_things),$$
$$\quad \text{not } ab(X, pounding\_tool).$$

$$capability(X, fire\_starting\_tool) \leftarrow$$
$$inst(X, wooden\_things),$$
$$\text{not } ab(X, fire\_starting\_tool).$$
$$capability(X, symbol) \leftarrow$$
$$inst(X, drawings),$$
$$\text{not } ab(X, symbol).$$

(The first rule above says that things with perforations can be used for focusing). The mentioned methodology allows us to express exceptions to these defaults, for instance

$$ab(X, pounding\_tool) \leftarrow$$
$$inst(X, glass\_things).$$

The above is a *weak exception* making inapplicable the default about the role of $pounding\_tool$. It says that things made out of glass, although hard, cannot be appropriately used as tools for pounding on other objects.

World Dynamics

ASP has established methodologies for representing cause and effect relations in what are called *discrete dynamic domains*. Theoretically, a discrete dynamic domain can be represented by a transition diagram whose nodes represent physical states of the domain and whose arcs are labeled by actions. A transition from a state σ_0 to a state σ_1 through an arc labeled by action a says that the execution of action a in σ_0 may take the domain to σ_1.

The modeling process of a dynamic domain in ASP starts with considering the types of objects of the domain, its relevant properties, and its possible actions. Domain properties are divided into *statics* (those properties that cannot be changed by actions) and *fluents* (those that can). Fluents are further divided into *inertial* and *defined*. Inertial fluents obey the default known as the Inertia Axiom, stating that *Normally things stay as they are* (McCarthy and Hayes 1969). Defined fluents are fluents defined in terms of other fluents; they can be seen simply as a convenient shorthand.

Let us see an application of this methodology tailored to our purposes. We begin by expanding the signature of $\mathcal{K}$ by a sort called $step$; a variable I ranging over steps; a relation $holds(F, I)$ (fluent F holds at step I); and a relation $occurs(A, I)$ (action A occurs at step I). We first consider the spatial layout of our scenarios. For simplicity, we only target here a very simple layout consisting of a sequence of adjacent areas. To model it, we add to our signature the static $next(A_1, A_2)$ (the next area after A_1 in the sequence is A_2). We are also interested in creating a basic formalization of motion in this geography. For that, we introduce the inertial fluent $at(X, A)$ (entity X is at area A) and the action $go\_to(X, A)$ (X goes to area A). We encode the *direct effect* of action go_to by the rule

$$holds(at(X, A), I + 1) \leftarrow$$
$$occurs(go\_to(X, A), I).$$

(i.e., if X goes to area A, he will be there in the next time step). *Indirect effects* of action go_to are represented by the axiom

$$\neg holds(at(X, A_1), I) \leftarrow$$
$$holds(at(X, A_2), I),$$
$$A_1 \neq A_2.$$

saying that an object cannot be at two areas simultaneously (and hence will no longer be where it used to be before going to some other area). Finally, *preconditions for the execution of an action* are encoded by rules of the type:

$$\neg occurs(go\_to(X, A), I) \leftarrow$$
$$holds(at(X, A), I).$$
$$\neg occurs(go\_to(X, A), I) \leftarrow$$
$$holds(at(X, A_1), I),$$
$$\neg next(A, A_1),$$
$$\neg next(A_1, A).$$

The informal reading is that one cannot go to an area where he already is, and that movement is only permitted between adjacent areas. In the latter rule above, note that we do not use the time step I for the static $next$, as its values do not change from one time step to another.

Now let us see how the same methodology can be expanded to the domain of perceiving and using objects. The signature of $\mathcal{K}$ will contain the action $use(X, Y, Z)$ (person X uses Y according to role Z). The following inertial fluents become relevant: $has\_used(X, Y)$ (person X has used object Y); and $orientation(X, Y)$ (the orientation of X is Y, where Y may be pos–the direction following the sequence of adjacent areas–or neg–opposite direction). Additionally there will be several inertial fluents describing the particular effects of using various objects according to different roles; one such example is $has\_observed\_new\_things(X)$ connected to the use of an object as a tool for focusing attention. The signature will also contain defined fluents $within\_sight(X, Y)$ and $within\_reach(X, Y)$ (X is within sight/ reach of Y). $\mathcal{K}$ will be expanded by the following axioms describing *direct effects of actions*:

$$holds(orientation(X, pos), I + 1) \leftarrow$$
$$occurs(go\_to(X, A), I),$$
$$holds(at(X, A_1), I), \quad A - A_1 \geq 0.$$
$$holds(orientation(X, neg), I + 1) \leftarrow$$
$$occurs(go\_to(X, A), I),$$
$$holds(at(X, A_1), I), \quad A - A_1 < 0.$$
$$holds(has\_used(X, Y), I + 1) \leftarrow$$
$$occurs(use(X, Y, Z), I).$$
$$holds(has\_observed\_new\_things(X), I + 1) \leftarrow$$
$$occurs(use(X, Y, focusing\_tool), I).$$

and similar rules for action use with other roles. $\mathcal{K}$ will include rules describing *indirect effects of actions* in terms of the newly added fluents:

$$holds(within\_sight(X, Y), I) \leftarrow$$
$$holds(at(Y, A_1), I),$$
$$holds(at(X, A_2), I),$$
$$holds(orientation(Y, pos), I), \quad A_2 - A_1 \geq 0.$$
$$holds(within\_sight(X, Y), I) \leftarrow$$
$$holds(at(Y, A_1), I),$$
$$holds(at(X, A_2), I),$$
$$holds(orientation(Y, neg), I), \quad A_2 - A_1 \leq 0.$$
$$holds(within\_reach(X, Y), I) \leftarrow$$
$$holds(at(X, A), I),$$
$$holds(at(Y, A), I).$$
$$holds(within\_reach(X, Y), I) \leftarrow$$
$$holds(within\_sight(X, Y), I),$$
$$instance(X, drawings).$$

And finally, $\mathcal{K}$ will specify *preconditions for the execution* of action use:

$$\neg occurs(use(X,Y,Z),I) \leftarrow \\ \quad \neg holds(within\_reach(Y,X),I).$$
$$\neg occurs(use(X,Y,pounding\_tool),I) \leftarrow fixed(Y).$$

where $fixed(X)$ is a static. Note that, in reality, $\Sigma(\mathcal{K})$ is supposed to include many more fluents and actions than the ones listed here, as well as a large set of axioms defining the cause-effect interdependencies between them.

For every inertial fluent f in the signature, $\mathcal{K}$ contains two defaults expressing the Inertia Axiom:

$$\begin{array}{rcl} holds(f,I+1) & \leftarrow & holds(f,I), \\ & & \text{not } \neg holds(f,I+1). \\ \neg holds(f,I+1) & \leftarrow & \neg holds(f,I), \\ & & \text{not } holds(f,I+1). \end{array}$$

For every defined fluent f in the signature, $\mathcal{K}$ contains:

$$\neg holds(f,I) \quad \leftarrow \quad \text{not } holds(f,I).$$

Basic Psychological Knowledge

The final component of $\mathcal{K}$ is an encoding of basic knowledge about cause-effect relationships from the *psychological* or *sociological* point of view, as well as information about the default behavior of different categories of people. To model this type of knowledge, we (1) use the methodology of representing defaults and their exceptions discussed earlier; (2) adopt and adapt the methods used in formalizing *physical* cause-effect relationships ; and (3) make use of the ASP formalization of a "theory of intentions" (Baral and Gelfond 2005; Gelfond 2006), as we explain later in this section.

One default statement that we may want to represent is: *Generally children are curious*. We add to $\Sigma(\mathcal{K})$ a static $curious(X)$, where X ranges over people. The default is then encoded using common ASP practices by:

$$\begin{array}{rcl} curious(X) & \leftarrow & instance(X,children), \\ & & \text{not } ab(X,curiosity). \end{array}$$

Commonsense knowledge may also tell us that: *Typically, when a curious person sees a misplaced object that he has not used yet, he becomes interested in that object.* To model this information, we need to add to $\Sigma(\mathcal{K})$ the static $misplaced(X)$ and the inertial fluent $interested\_in(X,Y)$ (person X is interested in thing Y). We then extend $\mathcal{K}$ by:

$$\begin{array}{l} holds(interested\_in(X,Y),I) \leftarrow \\ \quad curious(X), \ \ misplaced(Y), \\ \quad holds(within\_sight(Y,X),I), \\ \quad \neg holds(has\_used(X,Y),I). \\ \quad \text{not } ab(X,Y,interested\_in). \end{array}$$

Note that, since we now talk about the *default behavior* of people, this axiom combines the standard way of encoding *physical* indirect effects of actions with the methodology for representing defaults.

A slightly different type of knowledge that we may want to encode is the one captured by the sentence: *Normally, a person's interest in an object makes that person want to approach the object and use it according to its capabilities.* This phrase talks about what people intend to do, and about the triggering of such intentions. This is where the mentioned ASP *theory of intentions* (Baral and Gelfond 2005; Gelfond 2006) becomes useful. The theory consists of several ASP rules (that will now be part of program $\mathcal{K}$). Its main tenets are: *"Normally intended actions are executed the moment such execution becomes possible"* (non-procrastination):

$$\begin{array}{rcl} occurs(A,I) & \leftarrow & intend(A,I), \\ & & \text{not } \neg occurs(A,I), \\ & & action(A). \end{array}$$

and *"Unfulfilled intentions persist"* (persistence):

$$\begin{array}{rcl} intend(A,I+1) & \leftarrow & intend(A,I), \\ & & occurs(A,I), \\ & & action(A), \\ & & \text{not } \neg intend(A,I+1). \end{array}$$

We can use the new relation $intend(X,I)$ to express our default statement by way of the following two rules:

$$\begin{array}{l} intend(go\_to(X,A),I) \leftarrow \\ \quad holds(interested\_in(X,Y),I), \\ \quad \neg holds(within\_reach(Y,X),I), \\ \quad holds(at(Y,A),I), \\ \quad \text{not } ab(X,Y,interest). \\ 1\ \{\ intend(use(X,Y,Z),I) : capability(Y,Z)\ \}\ 1 \leftarrow \\ \quad holds(interested\_in(X,Y),I), \\ \quad holds(within\_reach(Y,X),I), \\ \quad \text{not } ab(X,Y,use). \end{array}$$

Intuitively, the second rule says that, if X is interested in Y and can reach it, then X will choose one of the possible uses of Y, and intend to use Y according to this role. Other defaults can be represented in a similar way, for instance: *Normally, humans mimic the behavior of other people*; *Typically, people do not intend to perform actions that are obviously impossible*; *Generally, people do not intend to perform actions that are punishable by law* (here, we would add a new static, $enforced(L)$, to say that a law is enforced), etc.

Finally, it is relevant to model changes in the emotional state of a person as an effect of using an object. This is done by adding the inertial fluent $emotional\_state(X,Y)$ (the emotional state of person X is Y, where Y has one of the values $unhappy$, $neutral$, or $happy$), and rules like:

$$\begin{array}{l} holds(emotional\_state(X,happy),I) \leftarrow \\ \quad holds(has\_observed\_new\_things(X),I), \\ \quad \text{not } ab(X,emotional\_state). \end{array}$$

saying that, normally, observing new things makes people happy. This concludes the description of the methodology of creating the background knowledge base $\mathcal{K}$.

Formalization of *Shikake* Principles

There are four criteria for determining whether a design qualifies as a *Shikake*: low expertise, low cost, long term continuous behavior changes, and a wide range of target users (Matsumura 2012). The first one, low expertise, was addressed by creating a *commonsense* background knowledge base containing only general knowledge about the

world and its dynamics, and the default behavior of people. Let us see how the other requirements are captured by a logic program, $\mathcal{P}$, whose signature contains the predicate $potential\_shikake(X, Y)$ (object X under consideration has the potential to work as a *Shikake* for person Y).

To meet the low cost requirement, the description of a design will have to be accompanied by cost information, given in terms of relation $cost(X, Y)$ (X's cost is Y, where Y ranges on a discrete scale from 1 to 5, with 1 meaning a very low cost and 5 a very high cost). $\mathcal{P}$ will contain the rules:

$$potential\_shikake(X, Y) \leftarrow cost(X, Z),\ Z \leq 2.$$
$$\neg potential\_shikake(X, Y) \leftarrow cost(X, Z),\ Z > 2.$$

Shikakeology assumes that long term continuous behavior changes are achieved "if people desire and enjoy changing their behavior" (Matsumura 2012). We formulate this idea via the rule:

$$\begin{aligned} potential\_shikake(X, Y) \leftarrow\ & \\ & \text{not } \neg potential\_shikake(X, Y), \\ & holds(emotional\_state(X, Y), 0),\ Y \neq happy, \\ & holds(emotional\_state(X, happy), n), \\ & has\_performed\_intended\_actions(X). \end{aligned}$$

where $has\_performed\_intended\_actions(X)$ means that person X has done all the actions that the designers of object Y intended him to do, and n is a constant denoting the final step. The rule says that a designed object has the potential of acting as a *Shikake* for a certain person if using the object according to its intended use makes that person happy.

Formulating this principle in precise terms in ASP raises the question of whether a designed object may be qualified as a Shikake *if a person becomes* unhappy *every time he does* not *perform the actions intended by designers to be performed in association with that object.* For instance, in the Tiny Shrine Gate example, if a person dumps garbage on the street and then observes the tiny shrine gate symbol, which makes him feel ashamed and therefore unhappy, would that be a successful *Shikake* scenario? This is just one example of how modeling *Shikake* principles in accurate terms may help clarify or move forward its definition.

The last requirement for an example to qualify as a *Shikake* is whether it targets a wide variety of users or not. To test this principle, we create scenarios about many different combinations of possible users (e.g., children or adults). A designed object is then considered a *Shikake* according to an analysis of the models of all considered scenarios, which we will describe in our presentation of system SHASP.

Encoding Design Examples and Scenarios

In this section, we show how a design example together with an associated target user scenario can be represented in ASP. We begin by considering the "Tennouji Zoo" example in the introduction and one target user, a child. This defines an instance of our inheritance hierarchy H that we specify in a program $\mathcal{S}_{zoo}$ via the facts:

$$is\_a(wooden\_cylinder, wooden\_things).$$
$$is\_a(wooden\_cylinder, things\_with\_perforations).$$
$$is\_a(c, children). \quad is\_a(zoo, public\_locations).$$

Let us consider a simple spatial layout consisting of a sequence of three adjacent areas: 1, 2, and 3. We know that:

$$misplaced(wooden\_cylinder).$$
$$fixed(wooden\_cylinder).$$
$$cost(wooden\_cylinder, 1).$$
$$enforced(fire\_starting\_law).$$
$$holds(at(wooden\_cylinder, 2), 0).$$
$$holds(at(c, 1), 0).$$
$$holds(emotional\_state(c, neutral), 0).$$

We assume that no other properties are true in the initial situation. This information is added to $\mathcal{S}_{zoo}$. It is known that the child wants to perform the sequence of actions $s = [go\_to(c, 2), go\_to(c, 3)]$, so $\mathcal{S}_{zoo}$ contains the fact:

$$intend(s, 0).$$

The purpose of the wooden cylinder is to cause people to use it as a focusing tool:

$$intended\_action(use(c, wooden\_cylinder, focusing\_tool)).$$

We can now check whether the wooden cylinder design qualifies as a *Shikake* for the given target user. For that, we assemble program $\mathcal{K} \cup \mathcal{P} \cup \mathcal{S}_{zoo}$, and compute its answer sets using the ASP solver CLASP. This program has only one answer set, which contains as expected:

$$occurs(go\_to(c, 2), 0).$$
$$occurs(use(c, wooden\_cylinder, focusing\_tool), 1).$$
$$occurs(go\_to(c, 3), 2).$$

Note that the child did not use the wooden cylinder as a pounding tool because it is fixed, and he did not use it for starting a fire because that is prohibited by an enforced law. Additionally, the answer set contains the atom

$$potential\_shikake(wooden\_cylinder, c)$$

saying that, at least for this type of target user, the wooden cylinder is an example of *Shikake* design. Other scenarios with different target users can be encoded in a similar way.

Next, let us encode as an ASP program $\mathcal{S}_{shr}$ the "Tiny Shrine Gate" example with one target user, an adult, and one available garbage bin. We assume that $\mathcal{K}$ contains: information about actions $throw\_garbage\_to\_bin(X)$ and $dump\_garbage(X)$ and their relation to inertial fluent $has\_garbage(X)$; knowledge saying that using a drawing as a symbol means to interpret it according to its meaning; and the psychological effects of performing socially desirable/undesirable actions in holy places. The instance of hierarchy H particular to our example is encoded by the facts:

$$is\_a(ad, adults). \quad is\_a(gb, garbage\_bins).$$
$$is\_a(tiny\_shrine\_gate, drawings).$$
$$is\_a(back\_street, public\_locations).$$

As before, we consider a simple spatial layout consisting of a sequence of five adjacent areas: 1, 2, 3, 4, and 5. We know:

$$misplaced(tiny\_shrine\_gate).$$
$$meaning(tiny\_shrine\_gate, holiness).$$
$$cost(tiny\_shrine\_gate, 1).$$
$$holds(at(ad, 1), 0).$$
$$holds(at(gb, 2), 0).$$
$$holds(at(tiny\_shrine\_gate, 4), 0).$$
$$holds(orientation(ad, pos), 0).$$
$$holds(has\_garbage(ad), 0).$$

$$holds(emotional\_state(ad, neutral), 0).$$
$$intend(s(5), 0).$$

where $s(5)$ is the sequence of go_to actions getting ad to area 5, by moving one area at a time. Additionally, ad wants to end up in a state in which he no longer has garbage. The program $\mathcal{K} \cup \mathcal{P} \cup \mathcal{S}_{shr}$ has two answer sets. In one of them, the adult first interprets the shrine gate symbol and then throws the trash to the bin, while moving to his destination. The adult ends up happy and, in this answer set, the drawing is a potential *Shikake*. In the second one, the adult dumps the garbage on the street while interpreting the shrine gate drawing in the same step. He ends up unhappy in the end, because he realizes that he performed an undesirable action in a holy place. *Based on current principles, this scenario does not support the idea that the design is a* Shikake.

Finally let us consider a modification of the "Tiny Shrine Gate" scenario where the drawing is absent, but we have an enforced law against dumping garbage in public locations instead. We know that the cost of enforcing this law is high. We represent this scenario by a program $\mathcal{S}_{law}$ replacing facts of $\mathcal{S}_{shr}$ about the shrine gate by:

$$enforced(garbage\_dumping\_law).$$
$$cost(garbage\_dumping\_law, 4).$$

We want to know whether this law can be considered a *Shikake*. The program $\mathcal{K} \cup \mathcal{P} \cup \mathcal{S}_{law}$ has a unique answer set in which the adult performs the intended action of throwing the garbage to the bin, but without a positive change in emotional state. Due to the high cost as well, the conclusion of the answer set is that law enforcement is not a *Shikake*.

The SHASP Intelligent Agent

Our system, SHASP, is given a collection of encoded scenarios related to the same design. For each such encoding $\mathcal{S}$, it assembles the logic program $\mathcal{K} \cup \mathcal{P} \cup \mathcal{S}$ and computes its models using the off-the-shelf ASP solver CLASP. SHASP contains a small piece of procedural code written in Python. This code processes all of the answer sets for all examples (i.e., adds to all atoms in an answer set an extra parameter identifying that specific answer set). It then collects the resulting facts in an ASP program, Π, that also contains a rule stating that a design can be considered a *Shikake* if and only if it is a $potential\_shikake$ for at least 75% of the target users in all computed answer sets. Finally, SHASP computes the answer set of Π, which will contain the atom $shikake(x)$ is the design x satisfies the requirements of *Shikake* design.

Conclusions and Future Work

In this paper we have shown how the *Shikake* approach can be encoded in ASP, and how this formalization can be used to determine the *Shikake* status of various examples. We have illustrated the use of a precise language like ASP in finding opportunities for further theory improvement.

In the future, this work can be extended by automatically creating multiple possible scenarios for the same design, so that the scenarios are substantially different from each other. Considering (all possible) distinct scenarios for a particular example will ensure its more reliable qualification as a *Shikake*. We would also like to explore how system SHASP can be expanded with capabilities for an automated generation of *Shikakes*. This will require extending the background knowledge base with substantial information about different classes of objects and their possible instances, and about effects and preconditions of actions. Once this is done, the task of generating a *Shikake* will be accomplished by a choice rule that will decide what object to "misplace" at what location. This follows the lines of standard methodologies in the ASP planning community.

References

Balduccini, M., and Gelfond, M. 2003. Diagnostic Reasoning with A-Prolog. *TPLP* 3(4–5):425–461.

Balduccini, M., and Girotto, S. 2010. Formalization of Psychological Knowledge in Answer Set Programming and its Application. *Theory and Practice of Logic Programming* 10(4–6):725–740.

Baral, C., and Gelfond, M. 1994. Logic Programming and Knowledge Representation. *Journal of Logic Programming* 19(20):73–148.

Baral, C., and Gelfond, M. 2005. Reasoning about Intended Actions. In *Proceedings of AAAI-05*, 689–694.

Baral, C. 2003. *Knowledge Representation, Reasoning, and Declarative Problem Solving*. Cambridge University Press.

Fan, J.; Barker, K.; Porter, B.; and Clark, P. 2001. Representing roles and purpose. In *Proceedings of the 1st International Conference on Knowledge Capture*, K-CAP '01, 38–43. New York, NY, USA: ACM.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. Conflict-Driven Answer Set Solving. In *IJCAI-07: Proceedings of the 20th International Joint Conference on Artifical Intelligence*, 386–392. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Gelfond, M., and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *Proceedings of the International Conference on Logic Programming (ICLP'1988)*, 1070–1080.

Gelfond, M., and Lifschitz, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing* 9(3/4):365–386.

Gelfond, M., and Lifschitz, V. 1998. Action Languages. *Electronic Transactions on AI* 3(16):193–210.

Gelfond, M. 2006. Going Places – Notes on a Modular Development of Knowledge about Travel. AAAI 2006 Spring Symposium Series, 56–66.

Inclezan, D. 2012. Modeling a Theory of Second Language Acquisition in ASP. In Rosati, R., and Woltran, S., eds., *Proceedings of the 14th International Workshop on Non-Monotonic Reasoning (NMR'2012)*.

Matsumura, N. 2012. Shikakeology. Retrieved from the web page: http://shikakeology.org.

McCarthy, J., and Hayes, P. J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.