Interactive Model Expansion
in an Observable Environment*

CARBONNELLE PIERRE
KU Leuven

VENNEKENS JOOST
KU Leuven

BOGAERTS BART
Vrije Universiteit Brussel

DENECKER MARC
KU Leuven

Abstract

Many practical problems can be understood as the search for a state of affairs that extends a fixed partial state of affairs, the *environment*, while satisfying certain conditions that are formally specified. Such problems are found in, e.g., engineering, law or economics.

We study this class of problems in a context where some of the relevant information about the environment is not known by the user at the start of the search. During the search, the user may consider tentative solutions that make implicit hypotheses about these unknowns. To ensure that the solution is appropriate, these hypotheses must be verified by observing the environment. Furthermore, we assume that, in addition to knowledge of what constitutes a solution, knowledge of general laws of the environment is also present. We formally define partial solutions with enough verified facts to guarantee the existence of complete and appropriate solutions.

Additionally, we propose an interactive system to assist the user in their search by determining 1) which hypotheses implicit in a tentative solution must be verified in the environment, and 2) which observations can bring useful information for the search. We present an efficient method to over-approximate the set of relevant information, and evaluate our implementation.

KEYWORDS: Knowledge Representation, Man-Machine Interface, Configuration Problem.

1 Introduction

> Some things are in our control
> and others not.
> *Epictetus*

Many practical problems can be understood as the search for a state of affairs that extends a fixed partial state of affairs, the *environment*, while satisfying certain conditions

that are formally specified. Such problems are found in, e.g., engineering, law or economics. Consider, for instance, the buyer of a house who has to declare the purchase for tax purposes. The environment consists of the properties of the house, of the seller and of the sales contract: they are not under the control of the purchaser. This environment has to be extended with a registration containing decisions on, e.g., which tax rebate to claim. Such rebates are subject to conditions, and have consequences, stated by law.

The Knowledge Representation (KR) methodology to solve such problems is to design a vocabulary to represent the relevant objects and concepts in the states of affairs, and to express the conditions as a theory in a declarative KR language. The theory is the set of laws to be satisfied. When the environment is fully known, it can be expressed as a structure, and the search for a solution can be performed automatically by Model Expansion inference on the theory and the structure (Mitchell et al. 2006).

However, the information about the environment is rarely fully known at the start of the search. In our example, the buyer of the house does not know all the attributes of the house when they start their registration. In such cases, two approaches have been implicitly used: (i) all *relevant* information about the environment is assumed to be known at the start of the search so that a solution can be determined without making any hypothesis about the environment (Felfernig et al. 2014a), or, (ii) when this assumption is dropped, the model expansion mechanism will choose values for the unknown environmental symbols, and the user has the implicit responsibility to verify that these values are indeed correct in the environment. We are not aware of any method to assist the user in the verification task that this second approach requires. Thus, there is a danger that the verification is incomplete, and that the user accepts a solution making incorrect assumptions in their environment.

The primary goal of this work is to study the problem of model expansion in an unknown, but observable, environment in a precise and formal way. To achieve this goal, it is crucial that we make an *explicit distinction* between environmental and decision symbols in the system. In this setting, an important difference arises between an *unknown* environmental symbol, and an *undecided* decision symbol. In both cases, the value of the symbol is not available to the system, but in the first case, it exists but is unknown, while in the second case, it does not exist yet. Values assigned to symbols by the system must be handled accordingly. When the system assigns a value to an environmental symbol, the user must go observe if the assigned value is correct.

Notice that the value of some environmental symbols may not be known at the start of the search. For example, the buyer may not know whether a rebate has already been claimed for a renovation. If this information becomes relevant, the buyer will have to "observe it" in the environment, e.g., by asking the seller. Distinguishing environmental from decision symbols is unique to our system. When the system is aware of this distinction, it can assist the user by highlighting which observations still need to be made to be sure that the proposed solution conforms to reality, as well as inform them that the final configuration is indeed appropriate (i.e., that all the hypotheses have been verified by observing the environment).

Next to making a distinction between two types of symbols, we also make a distinction between the laws of possible environments and the laws of acceptable solutions. The laws of possible environments are guaranteed to hold independently of the decisions of the user; by contrast, the satisfaction of the laws of acceptable solutions depends

on the decision of the user. Importantly, unlike consequences of the laws of possible environments, environmental consequences derived from the laws of acceptable solutions must be observed. To see this, assume we extend our previous examples with assertions stating which county belongs to which states, and what decisions a buyer must make to obtain a tax rebate in each state. The first assertions are part of the laws of possible environments, while the second assertions are part of the laws of acceptable solutions. If the property is known to be in a county, it is safe to conclude from the first set of assertions that the property is also in the state of that county: it is not necessary to observe the state in which it is. By contrast, suppose that the buyer decides to claim a tax rebate available in only one state. By the second set of assertions, the system concludes that the property must be in that state, but this is not a safe conclusion. Indeed, it is clear that this conclusion must be verified by observing the environment, and if the property is observed to be in another state, the user must be informed that he cannot claim that tax rebate.

To further assist the user in their task, the system can also determine which observations can bring useful information for the search. Suppose that a lower tax rate can be obtained only if the property is energy efficient and will be used for housing. If the buyer does not plan to use it for housing, then the energy efficiency does not need to be known. To capture this kind of scenario, we introduce a notion of *relevance*: informally, a symbol is relevant when it is an essential part of a solution. Relevance is instrumental in reducing the number of observations and decisions the user has to make. This is especially valuable when such observations are costly. Past efforts to determine relevant symbols in the context of interactive configuration have used so-called *activation rules*, i.e., rules that determine when symbols become relevant. These activation rules have to be added manually by the knowledge engineer. Our approach eliminates the need for such rules, and automatically derives the relevance of symbols based on their occurrence in minimal solutions instead. Furthermore, we propose an approximate, but efficient, method to derive relevance automatically, using standard solvers.

The rest of this paper is organized as follows. In Section 2, we introduce the logic concepts we use in this paper. Next, we formalize model expansion in a two-theory setting: one describing valid solutions and another describing possible environments, with a clear distinction between environmental and decision symbols (Section 3). In Section 4 and 5, we formalize the concept and operation of an assistant for interactive model expansion in that setting. Then, we define partial solutions with enough observations and decisions to guarantee that complete and appropriate solutions exist (Section 6). We define relevant symbols (Sections 7), and present an efficient method to over-approximate the set of relevant symbols. Finally, we evaluate our implementation (Section 8) and compare it to related work (Section 9)

2 Preliminaries

We assume familiarity with syntax and semantics of propositional logic and first-order logic (FO). To keep the presentation light, we develop our theory for propositional logic, but, in fact, the concepts we define apply to all logics with a model semantics. Our examples use a fragment of first-order logic.

A vocabulary Σ is a set of symbols. A theory is a set of formulae constructed from these symbols using the logic connectives $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow$ (negation, conjunction, disjunction, material implication, equivalence). A partial (resp. total) structure of Σ is a partial (resp. total) function from Σ to $\{\mathbf{t}, \mathbf{f}\}$, giving a Boolean value (a.k.a. interpretation) to each symbol. The value of a symbol $\sigma \in \Sigma$ in a structure S is denoted by σ^S if it exists. In this case, we say that σ is interpreted by S, and that the *fact* "σ has the value denoted by σ^S" is true in S. The set of interpreted symbols of S is denoted by Σ_S. If S is a total structure of Σ, we can determine the truth-value of all well-formed expressions and theories e over Σ; the resulting truth-value is denoted by $[\![e]\!]^S$. A total structure S that makes every formula in T true is a *model* of T; then, T is *satisfied* by S.

We call a partial structure S *less precise than* S' (and write $S \leq_p S'$) if $\Sigma_S \subseteq \Sigma_{S'}$ and all facts true in S are also true in S', i.e., if S' assigns the same values as S to the interpreted symbols of S (it may interpret more symbols). In this case, we call S' an *expansion of S*. The set of total expansions of S is denoted $\overline{S}$. A partial structure S is *consistent with* T if it has a *model expansion* for T, i.e., a total expansion that satisfies T. It *entails* T if all its total expansions satisfy T. We say that S and S' are *disjoint* if $\Sigma_S \cap \Sigma_{S'} = \emptyset$. For two disjoint structures S and S', we define $S + S'$ to be the structure that maps symbols σ interpreted by S to σ^S and symbols interpreted by S' to $\sigma^{S'}$.

3 Split model expansion

We first consider the generic problem of (non-interactive) model expansion in a two-theory setting, using two vocabularies, when the environment is fully known. The first vocabulary, Σ_{env}, contains the <u>environmental symbols</u>. The particular environment faced by the user is described by a total structure S_{env} for Σ_{env}. The second vocabulary, Σ_{sol}, is disjoint from the first and contains the decision symbols. A <u>solution</u> is fully described by a total structure S_{sol} for Σ_{sol}. The union of the 2 vocabularies is denoted by Σ_{all}.

We assume to have a theory T_{env} over vocabulary Σ_{env} that contains the laws satisfied by any environment that could be faced by the user. Any structure S_{env} that describes a particular environment is a model of T_{env}. We also assume that the constraints on the solution to the problem faced by the user are formalized by a logic theory T_{sol} in the vocabulary Σ_{all}. This theory fully captures all knowledge about when a solution is valid in a given environment.

Definition 1
The *split model expansion problem* $\mathcal{SMX}$ is a problem that takes as input a tuple $(\Sigma_{env}, \Sigma_{sol}, S_{env}, T_{env}, T_{sol})$ where S_{env} is a total structure over Σ_{env}, T_{env} is a theory over Σ_{env} satisfied by S_{env}, and T_{sol} is a theory over $\Sigma_{env} \cup \Sigma_{sol}$.
A *total solution* for such a problem is a total structure S_{sol} over Σ_{sol} such that

$$[\![T_{sol}]\!]^{S_{env}+S_{sol}} = \mathbf{t}.$$

Note that T_{env} is not used in the definition of the total solution because we assume that the environment satisfies T_{env} already. Having T_{env} as an additional input is useful in the process of finding a solution when the environment S_{env} is not fully known, but observable, as discussed in the next sections.

Example 1

We consider a simplified tax legislation for the sale of real estate, defined as follows. The tax amount is 10 % of the sales price for standard registration with the authorities, but a lower tax rate can be obtained when the house is bought to be leased to tenants for a low rent, or when the seller is licensed to sell social housing.

The environmental symbols we use to describe the seller and the house are propositions; the decision symbols are nullary function symbols: the type of registration (whose value is either *Social*, *Modest* or *Other*) and the tax rate (an integer). The states of affairs are described by structures over these vocabularies:

$$\Sigma_{env} = \{SocialHousing, LicensedSeller, LowRent\}.$$
$$\Sigma_{sol} = \{RegistrationType, TaxRate\}.$$

The theory T_{env} about the possible environments states that a social housing is necessarily leased at a low rent.

$$SocialHousing \Rightarrow LowRent.$$

The legislation is described by T_{sol}, stating the prerequisites for the types of registration, and the corresponding tax rate (where equality is interpreted as in first order logic):

$$RegistrationType{=}Modest \Rightarrow LowRent.$$
$$RegistrationType{=}Social \Rightarrow LicensedSeller \wedge SocialHousing.$$
$$TaxRate{=}1 \Leftrightarrow RegistrationType{=}Social.$$
$$TaxRate{=}7 \Leftrightarrow RegistrationType{=}Modest.$$
$$TaxRate{=}10 \Leftrightarrow RegistrationType{=}Other.$$

A possible sale is described by $S_{env}= \{SocialHousing{=}\mathbf{t}, LicencedSeller{=}\mathbf{t}, LowRent{=}\mathbf{t}\}$. The split model expansion problem has 3 total solutions:

$$\{TaxRate{=}1, RegistrationType{=}Social\},$$
$$\{TaxRate{=}7, RegistrationType{=}Modest.\},\ \text{and}$$
$$\{TaxRate{=}10, RegistrationType{=}Other.\}.$$

An interactive assistant for this theory is available online.[1]

4 Interactive model expansion

Often, a problem has many acceptable solutions in a given environment. Instead of choosing a particular solution by automated model expansion (Mitchell et al. 2006), it is preferable that the automated system allow the user to explore the set of acceptable solutions. For example, a property tax legislation may allow the buyer to register the sales in different ways, as in Example 1, and the system should allow them to see the various prerequisites and consequences of each valid options, e.g., on the tax rate.

We call the two interacting agents `System` and `User`. `System` offers `User` the freedom to choose values for environmental and decision symbols while verifying the validity of the solution that `User` constructs with respect to the 2 theories, T_{env} and T_{sol}. Since a

---

[1] https://tinyurl.com/simpleRegistration

system is often designed for many users, and each user wants a solution for their own environment, we assume it is the task of `User` to communicate information about their environment to `System`. `User` may have additional knowledge that is unknown to `System`. Indeed, it is often the case that some tacit knowledge is not formalized in the theories, but well known by the user. Obviously, `System` cannot use this tacit knowledge; instead, `User` must use it in the appropriate way.

At each step of an interactive process, `User` has constructed two partial structures, S_{obs} and S_{dec}, assigning values to environmental and decision symbols respectively. Those values correspond to <u>obs</u>ervations and to (tentative) <u>dec</u>isions respectively.

Definition 2
A *state of the interactive process* is a pair of partial structures (S_{obs}, S_{dec}) such that S_{obs} (resp. S_{dec}) is a partial structure over Σ_{env} (resp. Σ_{sol}).

By contrast, in prior work where there is no distinction between environmental and decision symbols, the state is described by only one partial structure.

Although `System` only has the information that `User` has provided to it, it also knows T_{env} and T_{sol}; hence, it can verify that the state is consistent with T_{env} and T_{sol}:

Definition 3
A state (S_{obs}, S_{dec}) is *consistent* (with T_{env} and T_{sol}) iff there is an expansion of the state that satisfies both theories:

$$\exists S \in \overline{S_{obs} + S_{dec}} : [\![T_{env} \cup T_{sol}]\!]^S = \mathbf{t}$$

During the interactive process, decisions can be added and retracted. This leads to the following definition:

Definition 4
An *interactive process* for a problem $\mathcal{SMX}$ with input $(\Sigma_{env}, \Sigma_{sol}, S_{env}, T_{env}, T_{sol})$ is a sequence $(S_{obs}^i, S_{dec}^i)_{0 < i \leq n}$ of n consistent states, such that, at each step, either the set of observations or the set of decisions is made either more precise or less precise, i.e., for any $0 < i < n$:
- $(S_{obs}^i <_p S_{obs}^{i+1} \vee S_{obs}^{i+1} <_p S_{obs}^i) \wedge S_{dec}^i = S_{dec}^{i+1}$
- or, $S_{obs}^i = S_{obs}^{i+1} \wedge (S_{dec}^i <_p S_{dec}^{i+1} \vee S_{dec}^{i+1} <_p S_{dec}^i)$.

In the interactive process, `User` and `System` work together towards a solution as follows:
- The interaction starts with a pair of empty structures, (S_{obs}, S_{dec}). If this state is inconsistent with T_{env} and T_{sol}, `System` tells `User` that there is no solution to the split model expansion problem.
- `User` can make S_{obs} or S_{dec} more precise as long as the new state is consistent. `System` prevents `User` from changing their decisions S_{dec} to an inconsistent state because it would not bring them closer to a solution. If `User` needs to make S_{obs} more precise following an observation, and if this change makes the state inconsistent, `System` asks `User` to retract a decision first so that the new observation can then be added without making the state inconsistent. To facilitate this task, `System` shows the list of decisions that, if retracted, would allow the entry. This list is the list of interpretations in the $\leq_p$-minimal structure S less precise than S_{dec} that, together with S_{obs}, is inconsistent with $T_{env} \bigcup T_{sol} \bigcup O$, where O is the observation to be added. It can be obtained using unsat_core methods (Cimatti et al.

2007). By repeatedly making the state more precise while keeping it consistent, User eventually obtains a total solution.

- User can make S_{obs} or S_{dec} less precise, i.e., retract past observations and decisions. The resulting state is necessarily consistent. This allows User to explore the complete space of solutions in their environment.

Example 2
Returning to Example 1, assume that User first decides that *RegistrationType=Social*. The corresponding state $(\{\}, \{RegistrationType=Social\})$ is consistent. They proceed by observing ¬*SocialHousing*. This cannot be added to the state because *SocialHousing* is a prerequisite for a *Social* registration, per T_{sol}. User thus needs to retract their only decision (*RegistrationType=Social*) before entering their observation.

Suppose instead that User selects *RegistrationType=Social* followed by *LowRent*=**f**. The resulting state is consistent with T_{sol} in the absence of T_{env}. However, the state is inconsistent with T_{env} because a *Social* registration can only be done for a social housing, which is necessarily leased at a low rent. User must backtrack on their decision *RegistrationType=Social* before entering the observation that the housing is not leased at a low rent.

Note that the presence of T_{env} reduces the number of consistent states, and thus shortens the search for a solution.

5 Propagation

System further assists User by informing them of the logical consequences of what is known in the current state. Such knowledge is of great value to User, and is used in most interactive assistant tools (Van Hertum et al. 2017; Falkner et al. 2019). It reduces the number of decisions that User has to make, and shortens the interaction in a way analogous to how unit propagation accelerates the naive backtracking algorithm in propositional satisfiability checking. The consequences of what is known are determined by a computation called *propagation* (or *backbone* computation).

Definition 5
A partial structure S' is a *T-propagation* of a partial model S of T iff it is an expansion of S with the same set of model expansions for T as S, i.e.,
- $S \leq_p S'$
- $\forall M \in \overline{S} : (\llbracket T \rrbracket^M = \mathbf{t} \Rightarrow M \in \overline{S'})$

Structure S' expands S by selecting some of the uninterpreted symbols whose value is the same in all models of T that expand S, and assigning them that value. We call S' the *optimal propagation* of S with respect to T if it is the most precise T-propagation. Algorithms to compute propagation are discussed in the literature (e.g., Zhang et al. (2019)).

In our interactive assistant framework, a complication arises when propagation derives facts for environmental symbols using the theory of solutions. In general, there is no guarantee that these facts are true in the environment faced by the user. To address this issue, we first propagate using T_{env} only, and then using both T_{env} and T_{sol}. This will further demonstrate the benefits of the split theory setting.

Because T_{env} only concerns environmental symbols, the propagation of S_{obs} by T_{env} can assign new values only to environmental symbols. We denote the structure containing these new values by S_{obs}^e. The propagation of the state by both T_{env} and T_{sol}, i.e., by T_{all}, can assign new values to both environmental and decision symbols. We denote the structures containing these new values by S_{obs}^a and S_{dec}^a. The structures S_{obs}^e, S_{obs}^a and S_{dec}^a are disjoint. `System` displays these structures differently:

- S_{obs}^e: The new environmental facts obtained by propagation of S_{obs} using T_{env} are shown as consequences to `User`: they do not need to be verified by observing the environment. This is safe because the propagated facts are true in the environment of `User`, by the definition of T_{env}.
- S_{obs}^a: The new environmental facts obtained by propagation of the state using T_{all} are shown as prerequisites of his decisions, to be verified by observing the environment. Indeed, there is no guarantee that they are true in the environment faced by `User`. The observation of the environment may occur at a later stage of the interactive process. If the observation does match the value obtained by propagation, `User` adds it to the state S_{obs}; the state will remain consistent. If the observation does not match, adding it would make the state inconsistent: `User` must first retract a decision to allow the entry, as explained in Section 4.
- S_{dec}^a: The new decision facts obtained by propagation of the state using T_{all} are shown as consequences to `User`. `User` cannot just change such a value, since that would cause inconsistency. If `User` wants to update the propagated value, they first need to retract decisions that led to the propagation.

Example 3
Returning to Ex. 1, the optimal propagation of the state ($\{LowRent=\mathbf{f}$, $SocialHousing=\mathbf{f}\}$, $\{\}$) is $S_{env}^e = \{\}, S_{env}^a = \{\}$ and $S_{dec}^a = \{RegistrationType=Other$, $TaxRate=10\}$: no observation needs to be verified. On the other hand, the optimal propagation of ($\{\}, \{TaxRate = 7\}$) is $S_{env}^e = \{\}, S_{env}^a = \{LowRent=\mathbf{t}\}$ and $S_{dec}^a = \{RegistrationType=Modest\}$ where $\{LowRent=\mathbf{t}\}$ needs to be verified.

6 Partial solutions

We now consider other forms of acceptable solutions. For example, it may not be necessary to observe the environment fully. In addition, some decisions may be unnecessary, in the sense that they can be made arbitrarily without impacting the result. Finally, the user may postpone some decisions when they have the guarantee that a solution exists.

We define definite and contingent solutions. A *definite* solution entails the satisfaction of T_{sol}, assuming T_{env}:

Definition 6
A *definite solution* of the split model expansion problem $\mathcal{SMX}$ is a state (S_{obs}, S_{dec}) such that (i) all facts in S_{obs} are true in S_{env} and (ii) T_{sol} is satisfied by every expansion S_o of S_{obs} that satisfies T_{env} and by every expansion S_d of S_{dec},

$$\forall S_o \in \overline{S_{obs}} : [\![T_{env}]\!]^{S_o} = \mathbf{t} \Rightarrow \forall S_d \in \overline{S_{dec}} : [\![T_{sol}]\!]^{S_d + S_o} = \mathbf{t}$$

Notice that Definition 6 treats T_{env} and T_{sol} very differently. This is because even though we do not know the full structure S_{env}, we do know that it exists, that it is more

precise than S_{obs}, and that it satisfies T_{env}. For this reason, we only need to consider expansions of S_{obs} that satisfy T_{env}. For such expansions, it should be the case that no matter how we make our choices, we obtain a solution to the problem at hand.

Example 4
A definite solution for the simplified real estate problem without any prior observation is $(\{\}, \{RegistrationType=Other, TaxRate=10\})$. Indeed, any sale can be registered with type *Other* and a tax rate of 10.

Any consistent state can be expanded into a definite solution. Indeed, by definition, every consistent state can be expanded into a total solution, which is a definite solution. In practice, definite solutions are often partial, not total, structures. As soon as `User` has found a definite solution, they can stop making observations and decisions. Indeed, such a solution can always be expanded to construct total solutions of the problem, by further observing the environmental and making any decision for the remaining decision symbols.

In some cases, `User` only needs the guarantee that a solution is reachable in their environment, by making the appropriate remaining decisions in a later stage. Since they do not know their environment fully, establishing this guarantee requires considering every environment more precise than what is known:

Definition 7
A *contingent solution* of the split model expansion problem $\mathcal{SMX}$ is a state (S_{obs}, S_{dec}) such that (i) all facts in S_{obs} are true in S_{env} and (ii) T_{sol} is satisfied by at least one expansion S_d of S_{dec} for every expansion S_o of S_{obs} that satisfies T_{env}, i.e.,

$$\forall S_o \in \overline{S_{obs}} : (\llbracket T_{env} \rrbracket^{S_o} = \mathbf{t} \Rightarrow \exists S_d \in \overline{S_{dec}} : \llbracket T_{sol} \rrbracket^{S_d+S_o} = \mathbf{t})$$

Example 5
A contingent solution for the simplified real estate problem without any prior observation is $(\{\}, \{RegistrationType = Other\})$. Indeed, any sale can be registered with type *Other*. The tax rate can be computed later (in this case easily).

Complexity The complexity of deciding whether a state is a contingent or definite solution is much higher than checking whether it is consistent and total. In the context of propositional logic, checking for definite solution is Π_1^p-complete (coNP-complete) while checking for contingent solution is Π_2^p-complete. In Section "Implementation", we propose a cheap approximate method to solve this problem.

7 Relevance

We now turn to the question of determining which observations are relevant for the search. We have shown that uninterpreted symbols in a definite solution (S_{obs}, S_{dec}) are *irrelevant* (or "do-not-care") in the following sense: S_{dec} extended with *any* interpretation for these symbols will satisfy T_{sol} in *any* possible environment that expands S_{obs} while satisfying T_{env}. In this section, we extend the notion of relevance to any state.

Definition 8
A $\leq_p$-*minimal definite solution* is a definite solution that is not more precise than another definite solution. We denote the set of minimal definite solutions more precise than a state by $MinDef(S_{obs}, S_{dec})$.

A minimal contingent solution is similarly defined; the set is denoted by $MinCont(S_{obs}, S_{dec})$.

Definition 9
The symbol σ in Σ_{all} is *relevant* in the consistent state (S_{obs}, S_{dec}) if, and only if, it is interpreted in at least one $\leq_p$-minimal definite solution more precise than the state, i.e.,

$$\exists S \in MinDef(S_{obs}, S_{dec}) : \sigma \in \Sigma_S$$

By our definition, all symbols interpreted in the state are relevant. As soon as User enters a value for a relevant symbol, the state is updated and the list of relevant symbols can be re-computed.

Proposition 1
If there are no uninterpreted relevant symbols in a consistent state, the state is a definite solution.

Proof
Suppose (S_{obs}, S_{dec}) is a consistent state without uninterpreted relevant symbols. By the definition of consistency, there is a total solution (and hence, a definite) solution that is more precise than (S_{obs}, S_{dec}), meaning that the set of definite solutions that expand (S_{obs}, S_{dec}) is non-empty. As a consequence (using the fact that $\leq_p$ is a well-founded order on the set of states of a fixed and finite vocabulary), this set has at least one $\leq_p$-minimal element. Let (S'_{obs}, S'_{dec}) be such a $\leq_p$-minimal definite solution that expands (S_{obs}, S_{dec}). By Def. 9, all symbols on which (S'_{obs}, S'_{dec}) and (S_{obs}, S_{dec}) differ are relevant. Since, by our assumption, all relevant symbols are interpreted in (S_{obs}, S_{dec}) (and thus also in (S'_{obs}, S'_{dec})), the state and the definite solution are actually equal.   □

Proposition 2
If there are no uninterpreted relevant environmental symbols in a consistent state, the state is a contingent solution.

Proof
Similarly to the previous proof, suppose (S_{obs}, S_{dec}) is a consistent state without uninterpreted relevant environmental symbols. Let (S'_{obs}, S'_{dec}) be a $\leq_p$-minimal contingent solution that expands (S_{obs}, S_{dec}). By the definition, all symbols on which (S'_{obs}, S'_{dec}) and (S_{obs}, S_{dec}) differ are relevant. Since, by our assumption, all relevant environmental symbols are interpreted in (S_{obs}, S_{dec}) (and thus also in (S'_{obs}, S'_{dec})), S_{obs} and S'_{obs} are equal. Since the state differs from the contingent solution only by the decisions and is consistent, it is a contingent solution too.   □

These properties allow System to alert User when a definite (resp. contingent) solution has been found. In propositional logic, the problem of computing whether a symbol σ is relevant in some state is Σ_3^p-complete. While computing the set of relevant symbols

is computationally expensive, it can be approximated cheaply, as discussed in the next section.

Example 6
When {*LowRent*=**f**}, *LicensedSeller* is irrelevant. Indeed, *RegistrationType* cannot be *Social* nor *Modest*, and the only $\leq_p$-minimal definite solution is {*LowRent*=**f**, *TaxRate*=10, *RegistrationType*=*Other*}, in which *LicensedSeller* is uninterpreted.

8 Implementation and validation

We developed a decision support tool that embodies our proposal, called the Interactive Consultant. Given two vocabularies ($\Sigma_{env}, \Sigma_{sol}$) and two theories ($T_{env}$ and T_{sol}), the tool generates a user interface that allows the user to follow the proposed interactive process, with the benefit of computer-generated guidance. The tool supports theories in FO($\cdot$), i.e., in first-order logic extended with types, aggregates, linear arithmetic over infinite domains, and (possibly inductive) definitions.[2] It uses the IDP-Z3 reasoning engine for FO($\cdot$) (Carbonnelle et al. 2022)[3], which is based on the Z3 SMT solver (de Moura and Bjørner 2008). The tool and its source code are available online.[4]

The user enters an observation or a decision by clicking on buttons or entering values in fields. The interactive advisor 1) updates the state, 2) propagates the state using T_{env}, 2) propagates again using $T_{env} \wedge T_{sol}$, and 3) determines the relevant symbols. The user interface is then updated to reflect the attributes and state of symbols: environmental vs. decision symbol, relevant, propagated. When there are no *environmental* symbols that are both relevant and uninterpreted, the solution is known to be a contingent solution for the environment. When there is no symbol that is both relevant and uninterpreted, the solution is known to be a definite solution for the environment, and the user can stop observing and making decisions.

Because relevance, as defined in Section 7, is expensive to compute, we approximate it using a 2-step process: 1) we compute propagation in state (S_{obs}, S_{dec}) as discussed in Section 5 and create theory C consisting of the propagated literals;[5] 2) we replace atoms occurring in the grounded[6] version of T_{all} by their value in the state (including propagated values), if any, and we simplify the formula using the laws of logic and arithmetic, to obtain T_{all}^*. Then, the set of definite solutions that extend state (S_{obs}, S_{dec}) for theory $C \cup T_{all}^*$ is the set of such solutions for theory T_{all}, i.e., $C \cup T_{all}^*$ and T_{all} are equivalent in that state. The symbols that do not occur in theory $C \cup T_{all}^*$ cannot be relevant: indeed, changing their value cannot change the truth-value of T_{all} in the state. Thus, theory $C \cup T_{all}^*$ contains all the symbols that are relevant in the state; it may also contain some irrelevant symbols, depending on how thoroughly the propagation and simplification is performed, but this is acceptable in practice. Since Step 1 is computed for the reasons

---

[2] https://fo-dot.readthedocs.io/
[3] www.idp-z3.be
[4] https://interactive-consultant.idp-z3.be/ and https://gitlab.com/krr/IDP-Z3
[5] I.e., if the T_{all}-propagation contains {$p = \mathbf{t}, q = \mathbf{f}$}, theory C contains {$p.\neg q.$}.
[6] The grounded version is obtained by expanding quantifications and aggregates over their (finite) domain.

given in Section 5, the additional cost of computing relevance is only determined by Step 2, and can be very low and still be useful to the user.

Definitions stated in the theory require special care when determining relevance. *Definitions* specify a unique interpretation of a defined symbol, given the interpretation of its parameters. They take the form of an equivalence between the defined atom and a *definiens*, i.e., a logic formula that defines it. A defined atom can be made irrelevant by replacing it by its definiens in the theory: the transformed theory will be equivalent (except for the defined symbol), and the defined atom will not occur in it anymore. Yet, defined atoms are often relevant for the user as it abstracts complexity in a useful way. Thus, we distinguish definitions from the other logical sentences in the theory. We consider a ground atom relevant if, and only if, it is important for the user to know its value, or it occurs in a simplified logical sentence, or it occurs in the simplified definition of a ground atom previously considered relevant.

To validate our approach, we have configured the decision support tool for Example 1 (available online[7]). Our tests showed that the system effectively assists the user by correctly determining which hypotheses must be verified, and which observations are relevant.

We then performed experiments on the real estate legislation adapted from Deryck et al. (2019) (available online[8]). We simulate the user by a robot entering values in the user interface in a random order. We measured how many decisions and observations the robot has to make, using two approaches:

- using the traditional approach where the problem instance is fully known before model expansion; the user enters the observations before his decisions, and stops when he finds a total solution;
- using our approach; the user only enters relevant observations and decisions, until he finds a definite solution.

In the traditional approach, the user has to make an average of 26.4 entries to obtain a model;[9] with the approach we propose, the user only needs to make 11.6 entries to obtain a definite solution, a 56% reduction (see details in Appendix). This reduction in the user's workload is also observed in other realistic use cases (Aerts et al. 2022).

9 Related work

The configuration problem, i.e., the problem of configuring a complex system out of simpler components to satisfy customer requirements, is a special case of model expansion that has been studied, e.g., by Felfernig et al. (2014b). Several authors have discussed interactive tools to solve configuration problems (Hotz et al. 2014; Madsen 2003; Van Hertum et al. 2017; Falkner et al. 2019). Our work is orthogonal to these studies. Indeed, the explicit distinction we make between environmental and decision symbols, and between the theory of the environment and the theory of acceptable solutions, can be

---

[7] https://tinyurl.com/simpleRegistration
[8] https://tinyurl.com/BelgianRegistration
[9] Depending on the order of data entry, the value of some environmental symbols may be derived by propagation of T_{env}: in that case, the user does not have to enter a value for them.

combined with the ideas from earlier work. To illustrate this, our interactive tool supports the 8 reasoning tasks identified as useful by Van Hertum et al. (2017).

Several approaches for controlling the relevance of symbols in configuration problems have been proposed. In these approaches, the modeler has the responsibility to write so-called *activation rules* that explicitly specify when a symbol becomes active (as done by Mittal and Falkenhainer (1990); Bowen and Bahler (1991) in Dynamic CSPs), or even when an entire component becomes active (as done by Stumptner and Haselböck (1993) in Generative CSPs). Our approach does not need activation rules to derive relevance of symbols. Jansen et al. (2016) also propose an approach without activation rules, based on justification theory. Its implementation requires a custom solver (Deryck et al. 2019). By contrast, our implementation uses a standard SMT solver.

To facilitate decision-making in companies, the Object Management Group has published a widely accepted standard in the business process communities, called the Decision Model and Notation standard (DMN) (OMG 2021). One extension of DMN, called cDMN (Vandevelde et al. 2021), further increases the expressivity of the notation. Decision models in those two notations can be translated automatically into FO($\cdot$) (Vandevelde et al. 2021), allowing our Interactive Consultant to support decision-making in companies. These DMN representations would be enhanced by using two vocabularies and two theories, as we propose, further highlighting the relevance of our work.

Our two-theory setting has some similarity with distributed constraint satisfaction problems, i.e., problems where several agents, each with their own constraints, search for a solution that satisfies their constraints (Yokoo 2012). Here, the environment, System and User may be viewed as agents, each with their own set of constraints. In our case, however, the theory of the environment is used by another agent, the interactive assistant.

The concept of relevance has been discussed in a special issue of *Artificial Intelligence* in 1997 (Subramanian et al. 1997), focusing on autonomous systems with limited computational resources. Relevance has been applied to improve the performance of propositional satisfiability solvers (Jansen et al. 2016). Instead, we apply it to reduce the workload of the user, by showing them the observations and decisions that make them progress in their search of a solution.

10 Conclusion

In this paper, we have studied the problem of model expansion in an unknown, but observable, environment in a precise and formal way. It was crucial to make an *explicit* distinction between environmental and decision symbols, as well as between the laws of possible environments and the laws of acceptable solutions. Additionally, we have shown how to reduce, safely, the number of observations and decisions to be made by the user of an interactive assistant in such a setting.

Relevance is an interesting topic for future work. The partial solutions we describe entail the satisfaction of the whole theory of acceptable solutions, given the environment. Sometimes, the user is interested in ensuring that a more limited goal is entailed by their decisions. By determining relevance relative to a goal, the number of observations can be further reduced. Furthermore, one could investigate whether some observations and decisions are more relevant than others.

Our approach could be generalized to problems with multiple agents participating in a model expansion, each with their own goals and constraints, e.g., in Community-based Configuration problems (Felfernig et al. 2014a) and Distributed Constraint Satisfaction (Yokoo 2012).

11 Competing interests

The authors declare none.

References

Aerts, B., Deryck, M., and Vennekens, J. 2022. Knowledge-based decision support for machine component design: A case study. *Expert Syst. Appl.*, *187*, 115869.

Bowen, J. and Bahler, D. Conditional existence of variables in generalised constraint networks. In Dean, T. L. and McKeown, K. R., editors, *Proceedings of the 9th National Conference on Artificial Intelligence* 1991, pp. 215–220. AAAI Press.

Carbonnelle, P., Vandevelde, S., Vennekens, J., and Denecker, M. 2022. IDP-Z3: a reasoning engine for FO(.). *CoRR, abs/2202.00343.*

Cimatti, A., Griggio, A., and Sebastiani, R. A simple and flexible way of computing small unsatisfiable cores in SAT modulo theories. In Marques-Silva, J. and Sakallah, K. A., editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings* 2007, volume 4501 of *Lecture Notes in Computer Science*, pp. 334–339. Springer.

de Moura, L. M. and Bjørner, N. Z3: An efficient SMT solver. In Ramakrishnan, C. R. and Rehof, J., editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008* 2008, volume 4963 of *Lecture Notes in Computer Science*, pp. 337–340. Springer.

Deryck, M., Vennekens, J., Devriendt, J., and Marynissen, S. Legislation in the knowledge base paradigm: Interactive decision enactment for registration duties. In *13th IEEE International Conference on Semantic Computing, ICSC 2019, Newport Beach, CA, USA, January 30 - February 1, 2019* 2019, pp. 174–177. IEEE.

Falkner, A. A., Haselböck, A., Krames, G., Schenner, G., and Taupe, R. Constraint solver requirements for interactive configuration. In Hotz, L., Aldanondo, M., and Krebs, T., editors, *Proceedings of the 21st Configuration Workshop* 2019, volume 2467 of *CEUR Workshop Proceedings*, pp. 65–72.

Felfernig, A., Hotz, L., Bagley, C., and Tiihonen, J. Configuration-related research challenges. In Felfernig, A., Hotz, L., Bagley, C., and Tiihonen, J., editors, *Knowledge-based configuration: From research to business cases* 2014a, chapter 15, pp. 191–7*192. Newnes.

Felfernig, A., Hotz, L., Bagley, C., and Tiihonen, J. 2014b. *Knowledge-based configuration: From research to business cases*. Newnes.

Hotz, L., Felfernig, A., Stumptner, M., Ryabokon, A., Bagley, C., and Wolter, K. Configuration knowledge representation and reasoning. In Felfernig, A., Hotz, L., Bagley, C., and Tiihonen, J., editors, *Knowledge-based configuration: From research to business cases* 2014, chapter 6, pp. 41–72. Newnes.

Jansen, J., Bogaerts, B., Devriendt, J., Janssens, G., and Denecker, M. Relevance for SAT(ID). In Kambhampati, S., editor, *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016* 2016, pp. 596–602. IJCAI/AAAI Press.

Madsen, J. N. 2003. Methods for interactive constraint satisfaction. Master's thesis, Department of Computer Science, University of Copenhagen.

MITCHELL, D. G., TERNOVSKA, E., HACH, F., AND MOHEBALI, R. 2006. Model expansion as a framework for modelling and solving search problems. Technical Report TR 2006-24, Simon Fraser University, Canada.

MITTAL, S. AND FALKENHAINER, B. Dynamic constraint satisfaction problems. In SHROBE, H. E., DIETTERICH, T. G., AND SWARTOUT, W. R., editors, *Proceedings of the 8th National Conference on Artificial Intelligence* 1990, pp. 25–32. AAAI Press / The MIT Press.

OMG 2021. Decision Model and Notation (DMN). `https://www.omg.org/dmn/`. Accessed: 2021-02-22.

STUMPTNER, M. AND HASELBÖCK, A. A generative constraint formalism for configuration problems. In TORASSO, P., editor, *Proceedings of Advances in Artificial Intelligence* 1993, volume 728 of *Lecture Notes in Computer Science*, pp. 302–313. Springer.

SUBRAMANIAN, D., GREINER, R., AND PEARL, J. 1997. The relevance of relevance (editorial). *Artif. Intell.*, *97*, 1-2, 1–5.

VAN HERTUM, P., DASSEVILLE, I., JANSSENS, G., AND DENECKER, M. 2017. The KB paradigm and its application to interactive configuration. *Theory Pract. Log. Program.*, *17*, 1, 91–117.

VANDEVELDE, S., AERTS, B., AND VENNEKENS, J. 2021. Tackling the DM challenges with cDMN: A tight integration of DMN and constraint reasoning. *Theory and Practice of Logic Programming*,, 1–24.

YOKOO, M. 2012. *Distributed constraint satisfaction: Foundations of cooperation in multi-agent systems*. Springer Science & Business Media.

ZHANG, Y., PU, G., AND ZHANG, M. SMTBCF: efficient backbone computing for SMT formulas. In AMEUR, Y. A. AND QIN, S., editors, *Formal Methods and Software Engineering - 21st International Conference on Formal Engineering Methods, ICFEM 2019, Shenzhen, China, November 5-9, 2019, Proceedings* 2019, volume 11852 of *Lecture Notes in Computer Science*, pp. 36–51. Springer.

Appendix A Detailed Validation Report

The theory to represent the real estate tax legislation has 27 environmental symbols and two decision symbols. We created 5 problem instances by randomly choosing Σ_{env} structures that satisfy T_{env}.

To simulate the traditional approach, the robot enters the environmental data in a random order, and at each step, propagation by T_{env} is performed. If the value of an environmental symbol is derived by this T_{env}-propagation, the robot does not have to enter it. After all environmental data is entered, the robot chooses decision values randomly among those that remain possible, until he obtains a total solution.

To simulate the approach we propose, at each step, the robot fills in one uninterpreted symbol, chosen randomly among the relevant ones, until he obtains a definite solution. If the symbol is environmental, it enters the value given by the problem instance; if it is a decision symbol, it chooses a valid value at random. When the state becomes inconsistent, `User` retracts a conflicting decision at random.

Table A 1 shows that the workload is significantly reduced when the system determines which observations are relevant, and which observations must be made in the environment to ensure the solution is definite, as we propose.

Table A 1. *The number of observations and decisions is greatly reduced with our proposal.*

| | Count of Observations and Decisions | |
	Traditional	Our proposal
Sale 1	26	10
Sale 2	26	14
Sale 3	26	10
Sale 4	27	10
Sale 5	27	14
Average	26.4	11.6
Gain		-14.8 (- 56 %)