References

[Abe93] M. Aben: Formally Specifying Re-usable Knowledge Model Components, *Knowledge Acquisition*, 5, 1993.

[AFS96] J. Angele, D. Fensel, and R. Studer: Domain and Task Modelling in MIKE. In A. Sutcliffe et al. (eds.), *Domain Knowledge for Interactive System Design*, Chapman & Hall, 1996.

[AWS93] J. M. Akkermans, B. Wielinga, and A. TH. Schreiber: Steps in Constructing Problem-Solving Methods. In N. Aussenac et al. (eds.): *Knowledge-Acquisition for Knowledge-Based Systems*, Lecture Notes in AI, no 723, Springer-Verlag, 1993.

[BAT+91] T. Bylander, D. Allemang, M. C. Tanner, and J. R. Josephson: The Computational Complexity of Abduction, *Artificial Intelligence*, 49: 25-60, 1991.

[BeA] R. Benjamins and M. Aben: Structure-Preserving KBS Development through Reusable Libraries: a Case Study in Diagnosis. To appear in International Journal on Human-Computer Studies.

[BeG96] R. Benjamins and C. Pierret-Golbreich: Assumptions of Problem-Solving Method. In N. Shadbolt et al. (eds.), *Advances in Knowledge Acquisition*, Lecture Notes in Artificial Intelligence (LNAI), no 1076, Springer-Verlag, Berlin, 1996.

[Ben95] V. R. Benjamins: Problem Solving Methods for Diagnosis And Their Role in Knowledge Acquisition, *International Journal of Expert Systems: Research and Application*, 8(2):93—120, 1995.

[BFS96] R. Benjamins, D. Fensel, and R. Straatman: Assumptions of Problem-Solving Methods and Their Role in Knowledge Engineering. In *Proceedings of the 12. European Conference on Artificial Intelligence (ECAI-96)*, Budapest, August 12-16, 1996.

[BrV94] J. Breuker and W. Van de Velde (eds.): *The CommonKADS Library for Expertise Modelling*, IOS Press, Amsterdam, The Netherlands, 1994.

[Cha86] B. Chandrasekaran: Generic Tasks in Knowledge-based Reasoning: High-level Building Blocks for Expert System Design. *IEEE Expert*, 1(3): 23—30, 1986.

[Dav91] M. David et al. (eds.): *Second Generation Expert Systems*, Springer-Verlag, 1991.

[EST+95] H. Eriksson, Y. Shahar, S. W. Tu, A. R. Puerta, and M. A. Musen: Task Modeling with Reusable Problem-Solving Methods, *Artificial Intelligence*, 79(2):293—326, 1995.

[FeB96] D. Fensel and R. Benjamins: Assumptions in Model-based Diagnosis. In *Proceedings of the 10h Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW´96)*, Banff, Canada, November 9th - November 15th, 1996.

[Fen95a] D. Fensel: Assumptions and Limitations of a Problem-Solving Method: A Case Study. In *Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-Based System Workshop (KAW-95)*, Banff, Canada, February 26th - February 3th, 1995.

[Fen95b] D. Fensel: *The Knowledge Acquisition and Representation Language KARL*, Kluwer Academic Publ., Boston, 1995.

[FeG96] D. Fensel and R. Groenboom: MLPM: Defing a Semantics and Axiomatization for Specifying the Reasoning Process of Knowledge-based Systems. In *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI-96)*, Budapest, August 12-16.

[FeS96] D. Fensel und R. Straatman: The Essence of Problem-Solving Methods: Making Assumptions for Efficiency Reasons. In N. Shadbolt et al. (eds.), *Advances in Knowledge Acquisiiton, Lecture Notes in Artificial Intelligence (LNAI)*, no 1076, Springer-Verlag, Berlin, 1996.

[FSG+96] D. Fensel, A. Schönegge, R. Groenboom and B. Wielinga: Specification and Verification of Knowledge-Based Systems. In *Proceedings of the 10th Knowledge Acquisition Workshop (KAW´96)*, Banff, Canada, November 9-14, 1996.

[KlW87] J. de Kleer and B. C. Williams: Diagnosing Multiple Faults, *Artificial Intelligence*, 32:97-130, 1987.

[KMR92] J. de Kleer, K. Mackworth, and R. Reiter: Characterizing Diagnoses and Systems, *Artificial Intelligence*, 56, 1992.

[Mar88] S. Marcus (ed.). *Automating Knowledge Acquisition for Experts Systems*, Kluwer Academic Publisher, Boston, 1988.

[Neb96] B. Nebel: Artificial Intelligence: A Computational Perspective. In G. Brewka (ed.), *Essentials in Knowledge Representation*, Springer Verlag, 1996.

[OHS96] K. O'Hara and N. Shadbolt: The Thin End of the Wedge: Efficiency and the Generalized Directive Model Methodology. In N. Shadbolt (eds.), *Advances in Knowledge Acquisition*, LNAI 1076, Springer-Verlag, Berlin, 1996.

[Pup93] F. Puppe: *Systematic Introduction to Expert Systems: Knowledge Representation and Problem-Solving Methods*, Springer-Verlag, Berlin, 1993.

[Rei92] W. Reif: The KIV-System: Systematic Construction of Verified Software, *Proceedings of the 11th International Conference on Automated Deduction*, CADE-92, LNCS 607, Springer-Verlag, 1992.

[Ste90] L. Steels: Components of Expertise, *AI Magazine*, 11(2), 1990.

[SWA+94] A. TH. Schreiber, B. Wielinga, J. M. Akkermans, W. Van De Velde, and R. de Hoog: CommonKADS. A Comprehensive Methodology for KBS Development, *IEEE Expert*, 9(6):28—37, 1994.

[SWB93] A. Th. Schreiber, B. J. Wielinga, and J. A. Breuker (eds.): *KADS: A Principled Approach to Knowledge-Based System Development, vol 11 of Knowledge-Based Systems Book Series*, Academic Press, London, 1993.

[HTW+92] G. van Heijst, P. Terpstra, B. J. Wielinga and N. Shadbolt: Using Generalised Directive Models in Knowledge Acquisition. In T. Wetter et al. (eds.), *Current Developments in Knowledge Acquisition*, LNAI, Springer-Verlag, Berlin, 1992.

[vHB92] F. van Harmelen and J. Balder: $(ML)^2$: A Formal Language for KADS Conceptual Models, *Knowledge Acquisition*, 4(1), 1992.

[vdV88] W. van de Velde: Inference Structure as a Basis for Problem Solving. In *Proceedings of the 8th European Conference on Artificial Intelligence (ECAI-88)*, Munich, August 1-5, 1988.

parsimonious explanation. Following our approach we try to find such assumptions by analysing the open goals in partial proofs. Here it is natural to start a proof for the conjecture that the so far established assumptions (*preference, input*, and *select-criterion assumption*) imply the *better-successor assumption*, which we have shown to be minimal. This proof attempt with KIV is straightforward and gets stuck in the following subgoal:

$$H' \subset H \wedge expl(H) \subseteq expl(H') \rightarrow \exists H''. (H'' \subset H \wedge expl(H) \subseteq expl(H'') \wedge successor(H,H''))$$

In order to obtain a concise assumption, we strengthen this goal by demanding that the H'' can be chosen such that $H' \subseteq H''$.

$$H' \subset H \wedge expl(H) \subseteq expl(H') \rightarrow \exists H''. (H' \subseteq H'' \subset H \wedge expl(H) \subseteq expl(H'') \wedge successor(H,H''))$$

Cutting out all subformulas concerned with *expl* (which is a natural technique to get simple assumptions), leads to the *successor assumption.*

Successor assumption:

$$H' \subset H \rightarrow \exists H'' .(H' \subseteq H'' \subset H \wedge successor(H,H''))$$

Under this assumption the subgoal can be reduced to the following

Abduction-problem assumption:

$$H' \subseteq H'' \subset H \wedge expl(H) \subseteq expl(H') \rightarrow expl(H) \subseteq expl(H'')$$

Together with the *select-criterion assumption* we were now able to prove that *hill climbing* finds a complete and parsimonious explanation. The *select-criterion assumption* guarantees completeness. The *successor* and *abduction-problem assumption* form a *task-specific translation* of the generic *better-successor assumption* for guaranteeing parsimonity. Each of the three assumptions restricts different parts of the required knowledge: the *select-criterion*, the *successor* relation, and the *explanation* function (and therefore the preference).

4.7 Monotonic Abduction Problems

The *abduction-problem assumption* we found is still not very intuitive. So we continue in strengthening it in order to get a more concise (but less minimal) assumption. By cutting out all subformulas containing H'' (a very simple and similar technique as before) we can strengthen this assumption to

$$\neg (H' \subset H \wedge expl(H) \subseteq expl(H'))$$

This however, would be a very restrictive assumption. A superset H of a set of hypotheses H' cannot have the same explanatory power. We slightly weakened this assumption to

$$(*)\ \neg (H' \subset H \wedge expl(H) \subset expl(H'))$$

which says that a smaller set of hypotheses should not explain more observations. Assuming it and trying to prove that the *abduction-problem assumption* is an implication of it leads to the even stronger *monotonic-abduction-problem assumption* that implies

$$H' \subset H \rightarrow expl(H') \subseteq expl(H)$$

This assumption imply (*) and the *abduction-problem assumption and* shows an interesting link to work on complexity analysis on abduction. Abduction in its general definition is an intractable problem class. [BAT+91] analyse several subclasses of abduction for their complexity. They prove that with the *monotonic-abduction-problem assumptions* it is possible to find a complete and parsimonious explanation in polynomial time. This assumption requires that a superset H' of a hypothesis H explains also a superset of data:

The monotonic-abduction problems defines a natural subclass of abduction. For example [KMR92] examine their role in model-based diagnosis. This assumption holds for applications, where no knowledge that constrain fault behaviour of devices is provided or where this knowledge respects the *limited-knowledge-of-abnormal behaviour assumption*. This is used by [KlW87] as *minimal diagnosis hypothesis* to reduce the average-case effort of finding all parsimonious and complete explanations with GDE. A syntactical way to ensure this assumption (i.e., a strengthening of this assumption) is the restriction of the domain theory to Horn clauses constraining only the correct behaviour of devices, cf. [KMR92]. It is interesting to see how the very generic *better-successor assumption* transforms into such intuitive task-specific assumptions.

5 Conclusions

In [FeS96] we proposed the idea of characterising and developing PSMs by their underlying assumptions. However, the problem arose how to find such assumptions. In this paper, we present the idea of the failed proof and its implementation by an interactive theorem prover. We developed and adapted PSMs by introducing assumptions that close the gap between the task and the competence of a PSM. Some of the assumptions we reported in section 4 during the adaptation process of the search strategy *hill climbing* to abductive tasks, could be integrated into its definition to form a strong (i.e., task-specific) PSM for abductive tasks. An interesting aspect that arose during the paper is the derivation of intuitive task-specific assumptions from very generic ones using the ontological commitments of the task.

KIV has shown to be an excellent tool for our purpose. Its concepts of proof modularity and proof reuse made the development and adaptation process of PSM, that is highly iterative and reversive, tractable. The interactive theorem prover could be used to identify assumptions as open goals in partial proofs. However, more work has to be done to integrate conceptual models like the model of expertise of CommonKADS [SWA+94] directly into the generic module concept of KIV. Furthermore, proof tactics that make use of this conceptual model are goals of our current work.

- The *parsimonious* requirement defines an implicit preference between the objects. A subset of an object is preferred if it is still a complete hypothesis. However, this preference defines only a *partial* order.

In the following, we will discuss the consequences for *hill climbing* when applying it to this new task.

4.2 Reformulating the Task

Reformulating the abduction task as a task to find a global optimum, needs a preference relation. This preference can naturally be derived from the parsimonious requirement. We define

Preference assumption
$H \langle H' \leftrightarrow H' \subset H \wedge expl(H) \subseteq expl(H')$

and therefore get

$parsimonious(H) \leftrightarrow \neg \exists H'. H \langle H'$

Thus, an explanation is parsimonious if it is a global optimum with respect to this preference.

4.3 Adapting Hill-Climbing to Partial Orders

The preference defined above is only a *partial* order. As a consequence, we have to weaken the knowledge requirements of the PSM by skipping the totality of the preference relation (cf. Figure 3). Surprisingly, we could weaken this knowledge requirement without weakening the competence of the method. Apparently the set of knowledge requirements in the original definition of *hill climbing* in Figure 2 was not minimal. The totality assumption is not necessary to achieve the desired competence of finding a local optimum.

However, the totality assumption was necessary in our original implementation of *hill climbing*. The main new problem when relaxing the totality assumption and working with partial orders comes from the fact that a best successor may not be comparable with its predecessor. Our original operationalization ignored this problem because it relied on the totality assumption.[3] By applying our version of *hill climbing* to the new task we found the more generic operationalization of Figure 2 that has the same competence with less assumptions. That is, by reusing the method we could get rid of some very specific assumptions that were related with an original application but that were not necessary for guaranteeing the desired competence. Applying the modular structure of proofs in KIV, only some proofs had to be redone. Proof reusability and proof modularity are essential properties of KIV to make the proof process practicable given the high need of revision and iteration in developing and adapting PSMs.

Like the operationalization, we also have to modify the *better-successor assumption* for ensuring that *hill climbing* finds a global optimum in the case of partial orders:

better-successor assumption:
$\exists y .(successor(x,y) \wedge y \in input \wedge uf(x) \langle uf(y))$
$\vee (z \in input \rightarrow \neg uf(x) \langle uf(z))$

4.4 Expressing Best Explanations as Global Optima

We have to prove that each global optimum is a complete and parsimonious explanation. A new assumption about the input arose during proving that each global optimum is parsimonious. We require that the input contains all sets of hypotheses.

Input assumption: $input(H)$[4]

It remains to guarantee completeness. The empty set of hypotheses is parsimonious but in most cases not complete. Therefore, not every global optimum is a best explanation. However, we could prove with KIV that each global optimum that is preferred over a complete explanation is also complete.

Completeness lemma:

$complete(H') \wedge H' \langle H \rightarrow complete(H)$

This lemma follows directly from the definition of the preference in section 4.2.

4.5 Finding Complete Explanations with Hill Climbing

We have to ensure that a local optimum that is found by *hill climbing* is a complete explanation. This follows from the *completeness lemma* by additionally requiring that the selection step delivers a complete explanation.

Select-criterion assumption:
$H \in select\text{-}criterion \rightarrow complete(H)$

Again, we first received an open goal during the proof process with KIV. It is necessary to know that the local optimum found by *hill climbing* is equal to or preferred over the selected start object of the search process for proving that such a local optimum is a complete explanation. This property actually follows from the operational specification of *hill climbing* (cf. Figure 2) but it has to be added to the competence description of *hill climbing* (cf. Figure 3). The extended post condition (cf. Figure 3) of hill climbing is therefore:

post condition
$hill\text{-}climbing(input) \in input$
$\neg\exists o. (successor(hill\text{-}climbing(input),o)$
$\wedge\, o \in input \wedge uf(hill\text{-}climbing(input)) \langle uf(o))$
$\boldsymbol{\exists x .(x \in input \wedge x \in select\text{-}criterion \wedge}$
$\boldsymbol{(x = hill\text{-}climbing(input)}$
$\boldsymbol{\vee\, uf(x) \langle uf(hill\text{-}climbing(input))))}$

4.6 Finding Parsimonious Explanations with Hill Climbing

We already know that a global optimum is a parsimonious explanation. Assumptions that ensure that *hill climbing* finds a global optimum therefore ensure that *hill climbing* finds a

3. With the totality assumption, a best successor is always comparable with its predecessor because it is comparable with each object. The original operationalization selected arbitrary one of the best successors and compared them with its predecessor.

4. Because we are only interested in complete explanations we could weaken the assumption to: $complete(H) \rightarrow input(H)$.

the task definition by the PSMs competence.

The question remains whether the assumptions are minimal or whether one is implied by an other. Here, *minimality* means that the assumption is *necessary* to guarantee that the competence of the PSM implies the task, formally,

$$(PSM_{competece} \rightarrow Task) \rightarrow Assumption$$

An assumption *minimal in the logical sense* (i.e., necessary) has the clear advantage that it maximizes the circumstances under which it holds. However, besides logical minimality other aspects like cognitive minimality (effort in understanding an assumption) or computational minimality (effort in proving an assumption) should influence the choice of assumptions.

In fact, we have proven with KIV that the corrected *better-successor assumption* is a minimal assumption in the logical sense. It was also easy to prove with KIV that the *better-successor assumption* is weaker[2] than the *totally-connected assumption* but the former has the disadvantage that it is not only formulated in terms of domain knowledge but also in terms of the current case input. Therefore, whether this assumption holds cannot be proven statically independent from the actual input. In general, minimizing (i.e., weakening) of assumptions can be achieved by analysing their sufficiency proof with KIV and eliminating aspects that are not necessary for continuing the proof.

The two assumptions that we have introduced yet are rather trivial. This is a consequence of our simplistic case study. In the following section we will define a more complex refinement of the current task and PSM which will lead to more interesting assumptions.

```
competence hill climbing
  sorts
    object, uf-value, objects : set of object
  functions
    uf: object → uf-value
    input: objects
    hill-climbing : objects → object
  predicates
    〈 : uf-value × uf-value
    successor : object × object
  variables x,y,z : uf-value, o : object
  axioms
    input requirement
      ∃o. (o ∈ input ∧ o ∈ select-criterion)
    knowledge requirement
      ¬(x 〈 x)
       x 〈 y ∧ y 〈 z →x 〈 z
      x 〈 y ∨ y 〈 x ∨ x = y
    post condition
      hill-climbing(input) ∈ input
      ¬∃o. (successor(hill-climbing(input),o)
      ∧ o ∈ input ∧ uf(hill-climbing(input)) 〈 uf(o))
endcompetence
```

Fig. 3. The competence theory of *hill-climbing*.

[2]. *A* is *weaker* than *B* iff *B* |= *A*.

4 Using Hill Climbing to Solve a Class of Abduction Problems

In the following, we introduce a more complex task definition. We ask for explanations, where an explanation is a set of hypotheses that explains a set of observations. That is, we define a typical abductive problem. We show how the PSM of section 3 can be adapted stepwise to the newly defined task by weakening some assumptions and by adding new assumptions.

4.1 The Task of Finding Complete and Parsimonious Explanations

[BAT+91] analyse the computational complexity of abduction. They define an abduction problem by a set of input data that must be explained and a set of hypotheses that can be used to construct explanations. A *complete explanation* must explain all input data (i.e., *observations*) and a *parsimonious* explanation must be minimal (that is, no subset of it explains all *observations*). Figure 4 provides the task definition for our new example. The goal describes what a *best explanation* must fulfil. The input requirement ensures that there are *observations*.

The following differences arise when comparing our new task with the task of section 3 (compare Figure 1):

- What was regarded as an object in section 3 is now a set of objects (i.e., an explanation is a set of individual hypotheses).
- The set of objects from which the best one is chosen is fixed and not provided as input. However, a set of data that must be explained by a selected object now plays the role of the input. Only, complete hypotheses are possible candidates for a *best explanation*.

```
task complete and parsimonious explanation
  sorts
    datum, data : set of datum,
    hypothesis, hypotheses : set of hypothesis
  functions
    expl: hypotheses → data
    best-explanation: hypotheses
    observables: data
  predicates
    complete: hypotheses
    parsimonious: hypotheses
  variables
    x : datum
    H,H' : hypotheses
  axioms
    goal
      complete(best-explanation)
      parsimonious(best-explanation)
      complete(H) ↔ expl(H) = observables
      parsimonious(H) ↔
         ¬∃H'. (H' ⊂ H ∧ expl(H) ⊆ expl(H' ))
    input requirement
      ∃x . x ∈ observables
endtask
```

Fig. 4. The task definition for *abduction*.

```
task global optimum
  sorts
    object[1], uf-value, objects : set of object
  functions
    uf: object → uf-value
    global-optimum : object
    input: objects
  predicates
    $\langle$ : uf-value $\times$ uf-value
  variables x,y,z : uf-value, o : object
  axioms
    goal
      $global\text{-}optimum \in input$
      $\neg\exists o\,.(o \in input \wedge uf(global\text{-}optimum) \langle uf(o)))$
    input requirement
      $\exists o\,.\,o \in input$
    knowledge requirement
      $\neg(x \langle x)$
      $x \langle y \wedge y \langle z \rightarrow x \langle z$
      $x \langle y \vee y \langle x \vee x = y$
endtask
```

[1]. *object* is a finite type.

Fig. 1. The task definition *global optimum*.

for our running example. The goal describes what an optimum must fulfil. There must be no other object that has a higher value for the utility function *uf*. The requirements ensure that there is a non-empty input and that the domain knowledge provides a total order relationship (irreflexive, transitive, and total).

3.2 The Problem-Solving Method: Finding a Local Optimum

We decided to choose *hill climbing* for our example. *Hill-climbing* is a local search algorithm that stops when it has found a local optimum. The main new requirement on *domain knowledge* that is introduced by *hill-climbing* (and by other local search methods) is the existence of a *successor* relationship between the objects that is used to guide the local search process. The control flow is defined in Figure 2. The method works as follows: First, we select a start object. Then we recursively generate the successors of the current object, select the best successors and compare them with the current object. If we find a better successor we recursively repeat the generation step. Otherwise, we return the object, that does not have better successors. The functions *select-one-object*, *generate*, *select-the-bests,* and *select-one-better* correspond to elementary inference actions in CommonKADS [SWA+94]. The competence theory in Figure 3 states that *hill climbing* is able to find a *local* optimum of the given set of objects. With KIV, we proved that our *hill-climbing* algorithm in Figure 2 always terminates and that it actually has the competence to find a local optimum.

3.3 Constructing Assumptions to Close the Gap between Task and PSM

Linking a task definition with a PSM requires two activities. First, the different terminologies have to be related (we will not go into this aspect during the paper). Second, we have to relate the strength of the PSM with the desired goal of the task definition. Usually assumptions have to be introduced for this purpose (cf. [Fen95a], [FeS96]). The PSM *hill-climbing* has the competence to find a local optimum in a graph. The task under concern requires to select an global optimum.

A trivial assumption which ensures that *hill-climbing* finds a global optimum is to require that each object is directly connected with each object.

totally-connected assumption: $successor(x,y)$

In this case, *hill-climbing* collapses to a complete search in one step as all objects are successors of each possible start object. If we improve the selection of the start object (i.e., on *select criterion*) we could weaken the assumption to: Every starting object must be connected with all objects. The requirement on *select criterion* would be to provide such an object.

A less drastic assumption is to require that each object (except a global optimum) has a successor with a higher preference.

better-successor assumption:

$$\exists y\,.(successor(x,y) \wedge uf(x) \langle uf(y)) \\ \vee uf(x) = uf(global\text{-}optimum)$$

However, we realised during the proof process that this assumption is too weak to guarantee the implication of the task definition by the PSM´s competence. We got the open goal:

$y \in input$

That is, one has to add an assumption about the *input* of the task. The missing piece of the assumption was detected as a remaining open premise of an interactively constructed, partial proof with KIV that failed to show the implication of

```
output := hill-climbing(input)
hill-climbing(input)
begin
    current := select-one-object(input);
    output := hill-climbing-resursion(current)
end
hill-climbing-recursion(current)
begin
    successors := generate(current);
    if successors = $\emptyset$
    then output := current
    else
        best-successors := select-the-bests(successors)
        new := select-one-better(current,best-successors)
        if current = new
        then    output := current
        else    hill-climbing-recursion(new)
        endif
    endif
end

select-one-object(x) $\in$ x
```

Fig. 2. The control flow of *hill-climbing*.

proof, but they are often neither *necessary* for the proof nor *realistic* in the sense that application problems will fulfil them. Therefore, further work is necessary to find improved characterizations for these assumptions. This is achieved by a precise analysis of their role in the completed proof that is used to retrace unnecessary properties of them.

We provide tool support for this process by adapting the Karlsruhe Interactive Verifier (KIV) [Rei92] for our purpose. KIV was originally developed for the verification of procedural programs but it can be applied to formal specifications of PSMs in dynamic logic as used by $(ML)^2$ [vHB92], KARL [Fen95b], or MLPM [FeG96]. It is the interactive character of the underlying tactical theorem prover of KIV that makes it suitable for hunting hidden assumptions. Instead of returning with a failure KIV returns with open goals that could not be solved during its proof process. Further important support is provided by correctness management and reuse facilities. As the development process of the appropriate task definition, PSMs, and assumptions is an iterative and reversible process, one has to keep track of (repeated) changes of lemmas, assumptions and proofs.

The content of the paper is organised as follows. In section 2, a framework for the specification of PSMs is sketched. In section 3, we discuss our first example. We introduce the task *find a global optimum* and the local search technique *hill climbing* as PSM. Because *hill climbing* can only guarantee to find a local optimum, assumptions have to be introduced to bridge the gap between task and PSM. We then discuss in section 4, how the specification of the PSM and its assumptions must be modified when they get reused for a more complex task. The task is to *find a best explanation* where an explanation is a set of hypotheses. The definition of hill climbing has to be changed and the assumptions has to be refined.

2 A Framework for Specifying Problem-Solving Methods

We identify four different aspects of a specification of KBS: a *task definition* defines the problem to be solved by the KBS; a *problem-solving method* defines the reasoning process of the KBS; a *domain model* describes the domain knowledge of the KBS; and *adapters* that are necessary to adjust the reusable elements to each other and to the specific requirements of a given application problem.

The *task definition* specifies the *goals* that should be achieved in order to solve a given problem. A task definition also defines *assumptions* about the domain knowledge. For example, a task that concerns the selection of the maximal element of a set of elements, requires a preference relation as domain knowledge. Assumptions are used to define the requirements on such a relation (e.g. transitivity, symmetry, etc.).

The reasoning of a KBS can be described by a *problem-solving method* (*PSM*). A PSM consists of three parts. First, a definition of its *competence* (cf. [vdV88], [AWS93]). Second, an *operational description* which defines the dynamic reasoning process, i.e. describes how the competence can be achieved in terms of the reasoning steps and their dynamic interaction (i.e., the knowledge and control flow). The third part of a PSM concerns *assumptions* about the domain knowledge. Each inference step requires a specific type of domain knowledge with specific characteristics. Preconditions on inputs are complemented by complex requirements on available domain knowledge.

The description of the *domain model* introduces the domain knowledge as it is required by the PSM and the task definition. Three elements define a domain model. First, a meta-level description of properties of the domain knowledge. This is the counterpart of the assumptions on domain knowledge made by the other parts of a KBS specification. In the case of knowledge acquisition these properties define goals for the modelling process of domain knowledge. The second element of a domain model concerns the *domain knowledge* and *case data* necessary to define the task in the given application domain, and necessary to carry out the inference steps of the chosen problem-solving method. The third element is formed by *external assumptions* that link the domain knowledge with the actual domain (e.g., complete fault models, no measurement faults etc.).[1]

The description of an *adapter* maps the different terminologies of task definition, PSM, and domain model and introduces assumptions that have to be made to relate the competence of a PSM with the functionality defined by the task. It relates the parts of a specification to each other and establishes their relationship in a way that meets the specific application problem.

The assumptions of the different parts of the specification of a KBS define their proper relationships and the adequate relationship of the overall specification with its environment. During the following, we will provide several illustrations for these assumptions and a method to find and to construct them.

3 A Simple Case Study: Global Optimum and Hill Climbing

We take a very simple problem as starting point for illustrating our ideas. We investigate the task of finding a global optimum and use the weak problem-solving method (or search strategy) *hill climbing* to solve this problem (cf. [FSG+96]). Additional assumptions have to be provided by the adapter because *hill climbing* can only guarantee to find a local optimum. In the following, we sketch the different elements of a specification and describe the process of finding assumptions.

3.1 The Task: Finding an Optimum

The definition of the task introduces the goal and requirements on domain knowledge necessary to define the task for a given application. Figure 1 provides the definition

1. These external assumptions can be viewed as the missing pieces in the proof that the domain knowledge fulfils its meta-level characterisations.

Assumption Hunting as Developing Method for Problem-Solving Methods

Dieter Fensel[1] and Arno Schönegge[2]

[1] University of Karlsruhe, Institute AIFB, 76128 Karlsruhe, Germany. E-mail: dieter.fensel@aifb.uni-karlsruhe.de
[2] University of Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, 76128 Karlsruhe, Germany.
E-mail: schoeneg@ira.uka.de

In *Proceeedings of the Workshop on Problem-Solving Methods for Knowledge-based Systems at the 15th International Joint Conference on AI (IJCAI-97)*, Nagoya, Japan, August 23, 1997.

Abstract

Problem-solving methods (PSMs) for knowledge-based systems need to make assumptions to provide effective and efficient problem solving: assumptions about the scope of the problem they should solve and assumptions about the domain knowledge they can use as a resource for their reasoning process. If these assumptions are made explicit they can improve the reusability of PSMs by guiding the refinement process of problem-solving methods for a given application and by defining goals for the acquisition process of domain knowledge. However, making the underlying assumptions explicit is not an easy task. The goal of our paper is to contribute to solve this problem. The main idea is to construct mathematical proofs and analysis of their failure as a systematic means for formulating assumptions. Tool support is provided by adapting the Karlsruhe Interactive Verifier (KIV) for our purpose. KIV is an interactive theorem prover that returns with open goals if a proof could not be completed. These open goals can be used to derive the assumptions we are looking for.

1 Introduction

The concept *problem-solving method* (PSM) is present in a large part of current knowledge-engineering frameworks (e.g. GENERIC TASKS [Cha86]; ROLE-LIMITING METHODS [Mar88], [Pup93]; KADS [SWB93] and CommonKADS [SWA+94]; the METHOD-TO-TASK approach [EST+95]; COMPONENTS OF EXPERTISE [Ste90]; GDM [HTW+92]; MIKE [AFS96]). Libraries of PSM are described in [Ben95], [BrV94], [Cha86], and [Pup93]. In general, a PSM describes which reasoning steps and which types of knowledge are needed to perform a task. Such a description should be domain and implementation independent. PSMs are used in a number of ways in knowledge engineering: as a guideline to acquire problem-solving knowledge from an expert, as a description of the main rationale of the reasoning process of the expert and the knowledge-based system (KBS), as a skeletal description of the design model of the KBS, and to enable flexible reasoning by selecting methods during problem solving.

During the last years, PSMs have become quite successful in describing the reasoning behavior of KBS. However, there is still no solid theoretical background in characterising the precise competence of PSMs and in providing guidelines for developing reusable PSMs and for adapting these PSMs to application specific circumstances. Recent work tries to achieve both by characterising PSMs in terms of their underlying assumptions (cf. [Fen95a], [BeG96], [FeS96], [OHS96], [FSG+96], [BFS96], [FeB96], [BeA]).

In general, most problems tackled with KBSs are inherently complex and intractable (cf. [FeS96], [Neb96]). Efficient reasoning is only possible by introducing assumptions. These assumptions are necessary to reduce the complexity of the reasoning task and the development process of the reasoning system. They either formulate requirements on domain knowledge used by the PSM or restriction on the size of the problem that is solved by the PSM (cf. [BFS96]).

As a consequence the important question arises of *how to get such assumptions*. In this paper, we introduce a systematic approach for *constructing* such assumptions. We propose a method and a tool for this purpose. The main idea is to use mathematical proofs and analysis of their failure as a systematic means for forming assumptions. A mathematical proof that a PSM solves a given problem usually enforces the introduction of assumptions to close gaps in the line of reasoning of the proof. It can therefore be viewed as a search process for hidden assumptions. Gaps that can be found in a failed proof provide already first characterizations of these assumptions. An assumption that implies a lemma would close the gap in the proof is a possible candidate we are looking for. That is, formulating this lemma as an assumption is a first step in finding and/or constructing assumptions that are necessary to ensure that the competence of a PSM is strong enough to achieve the goals as they are defined by the task.

Using an open goal of a proof directly as an assumption normally leads to very strong assumptions. That is, these assumptions are *sufficient* to guarantee the correctness of the