Retaining Consistency in Temporal Knowledge Bases

Franz Wotawa and Bibiane Angerer*

Graz University of Technology, Institute for Software Technology,
8010 Graz, Inffeldgasse 16b/2, Austria
{wotawa, bangerer}@ist.tugraz.at
http://www.ist.tugraz.at/wotawa/

Abstract. Retaining consistency for large knowledge bases is a difficult task. This holds especially in the case where the knowledge base comprise temporal knowledge and where the knowledge comes from independent and unreliable sources. In this paper we propose the use of temporal logics, i.e., CTL, to describe the background theory and the corresponding Kripke Structure to store the temporal knowledge. Moreover, we introduce a declarative formalization of belief revision which is necessary to keep the knowledge base in a consistent state. Finally, we discuss how the structure of CTL formulas can be used to implement belief revision. The research described in the paper is motivated by a project that deals with automating the analysis of meetings, e.g., to provide meeting summaries, where cameras, microphones, and other sources of knowledge has to be integrated.

Keywords: Knowledge-processing, KBS methodology, temporal reasoning.

1 Introduction

The handling of information that comes from different independent sources is a difficult task. One reason for this observation is the fact that the given information is inconsistent because different sources deliver contradicting information. In computer science this problem becomes worst when considering the following situation. Given a set of sensors which are used for gaining information from a certain situation. Each of these sensors delivers a specific view on the situation. Because of reliability issues and different methodologies used the views are very likely to be inconsistent. Consider for example a meeting situation where several people are interacting (see Figure 1) together. During the meeting the participants present ideas, discuss them, and agree or disagree on results. Usually a meeting is more or less structured. Now assume that we want to have a system that allows for automatically analyzing a meeting with the purpose of providing a meeting summary or to allow for answering questions regarding the meeting

* Authors are listed in reverse alphabetical order.

M. Ali and R. Dapoigny (Eds.): IEA/AIE 2006, LNAI 4031, pp. 600–609, 2006.

and its content. Figure 2 presents the structure of such a system. The meeting analysis system (MAS) has two different sensing devices and a number of other information sources like meeting invitations and schedules, or presentation materials to be used during the meeting. The sensing devices are an audio device which delivers the audio signals together with estimations of the position of the audio source and a video system which allows for recognizing meeting participants and their positions. Moreover, we assume the existence of a system that converts audio data into a textual representation.

In order to provide meeting summaries or to answer given questions the MAS has to store all information gained from the sensing devices, the modules attached to the system, and available background knowledge. Because of unreliable sensors and conversion routines like speech-to-text the observed information is very unlikely to be consistent with the background knowledge. Hence, providing a consistent world model which should be the final result of the information integration module is difficult. Beside consistency we face the problem that the observed information is not complete with respect to the available information a human can gain when listening to the audio and watching the available video information.

The objective of this paper is to provide a knowledge representation schema together with reasoning mechanisms that allow for:

1. checking and ensuring consistency, and
2. preserving as much information as possible and gaining new one.

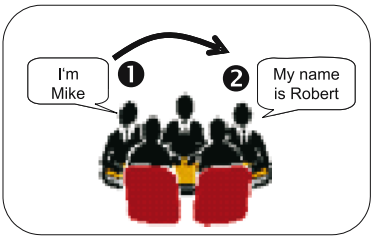

Fig. 1. A classical meeting situation

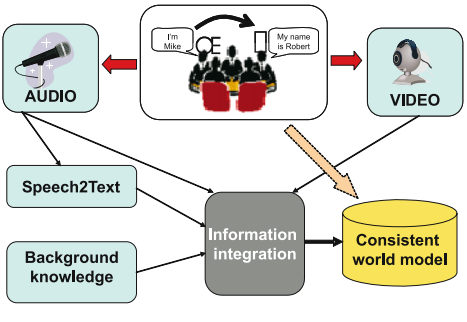

Fig. 2. Information sources

Based on this knowledge representation schema other applications like automated meeting analyzing tools are possible. Such applications require some sort of query language that enable the extraction of the needed information from the consistent knowledge base.

2 Representing the World

The information integration module (Fig. 2) obtains events and state information that represent the current state of a meeting from the attached sensors like audio or video. This information usually comprises a proposition like the number of participants which are detected by the video system together with some measure indicating the confidence about the proposition. Although the confidence can be high the proposition does not have to be true in the real world at the current point in time. Moreover, due to the fact that audio and video sensors exchange information directly they may deliver contradicting input to the information integration module. These contradictions which may not be detected at a certain state but later have to be identified and removed. The removal of contradictions can be done by removing propositions from world states such that the world state together with the underlying background theory are consistent again. Unfortunately, there can be several solutions for one problem and we have to deal with alternative world states in order to avoid loosing any information gathered by the sensors.

A knowledge representation schema that is capable to represent all information about an evolving situation like the mentioned meeting situation together with alternatives has to fulfill the following requirements:

1. Consider the storage of temporal knowledge.
2. Allow representation of background knowledge.
3. Handle alternative scenarios within the same schema.
4. Enable consistency checks.

Regarding background knowledge we want to specify sentences like the following.

"It cannot be the case that the number of participants changes over time when nobody is leaving the room."

"If a person is identified and the name is associated, then the name of the person does not change in future states."

Hence, the formalization of background knowledge itself requires to capture temporal aspects.

A knowledge representation schema that captures all requirements is the temporal logic Computation Tree Logic* (CTL*) [7]. CTL* (and in particular its fragments Linear Time Temporal Logic (LTL) and CTL) have been successfully used for verification of hardware and more recently software [4, 10]. For our purpose CTL can be used to formalize the background theory. The consistency check is done by checking the background theory against the current world model which is basically a state transition diagram. Each state comprises a set

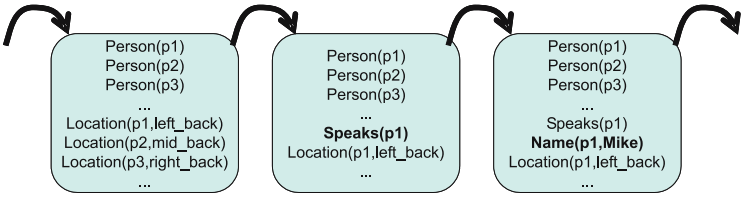

Fig. 3. Parts of a evolving world model

of proposition that are true. States are connected via directed edges. Going from one state to its successor state corresponds to advancing time. Each state corresponds to a time interval in the real world where the given set of propositions do not change. Figure 3 shows a part of a world model.

Formally, the world model is represented as a Kripke Structure $K = (A, S_0, S, R, L)$ where A is a finite set of atomic propositions, S is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $R \subseteq S \times S$ is a relation which represents the edges between the states, and $L : S \mapsto 2^A$ is a mapping assigning each state of S a set of atomic propositions true in that state. L is called label function. For example, the label function L for the leftmost state in Fig. 3 comprises the propositions $Person(p1), Location(p1, left_back)$ and others.

In order to be self contained we briefly introduce CTL and its semantics. In CTL we distinguish between state formulas and path formulas. Path formulas allow to speak about temporal issues whereas state formulas represent knowledge about one state. CTL formulas can use the branching time operators **A** and **E** which stand for every respectively some computational path and linear-time operators as well as operators from standard propositional logic. The linear time operators are **X** (next time), **U** (until), and **V** (unless). Formally, CTL formulas are state formulas which are inductively defined as follows:

1. Any atomic proposition $a \in A$ is a state formula.
2. If ϕ and ρ are state formulas, then $\neg\phi$, $\phi \wedge \rho$, and $\phi \vee \rho$ are state formulas.
3. If ϕ and ρ are state formulas, then $\mathbf{X}\phi$, $\phi\mathbf{U}\rho$, and $\phi\mathbf{V}\rho$ are path formulas.
4. If ϕ is a path formula, then $\mathbf{E}\phi$ and $\mathbf{A}\phi$ are state formulas.

Other operators like **F** (finally) and **G** (globally) can be expressed as $\mathbf{F}\phi \leftrightarrow$ **true U** ϕ and $\mathbf{G}\phi \leftrightarrow$ **false V** ϕ respectively.

For example, saying that a particular person cannot have different names during the meeting is represented by the following CTL formula:

$$AG\neg\,(Name(X,Y) \wedge Name(X,Z) \wedge Y \neq Z)$$

We also can express the fact that knowing a person's name in one state allows us to derive the person's name in the next state.

$$AG\,(Name(X,Y) \rightarrow XName(X,Y))$$

The previous CTL formula can be used to check consistency of different information sources. The latter one is for completing the knowledge base once some information is missing.

The semantics of CTL is expressed by the entailment relation $K, s \models \phi$ which means that formula ϕ is true in a state $s \in S$ of a Kripke Structure K. To formalize the entailment relation we first introduce the notation of paths. A path π of a Kripke Structure K is an infinite sequence of states $\langle s_0, s_1, s_2, \ldots \rangle$ such that each successive pair of states (s_i, s_{i+1}) is an element of R, i.e., there exists an edge between s_i and s_{i+1} in the Kripke Structure. $\pi(i)$ denotes the i-th element of π ($i \geq 0$), and π^j denotes a suffix of π starting at element j, i.e., $\pi^j = \langle \pi(j), \pi(j+1), \ldots \rangle$.

1. $K, s \models a$ if $a \in L(s)$, for any atomic proposition $a \in A$.
2. $K, s \models \neg\phi$ if $K, s \not\models \phi$
3. $K, s \models \phi \wedge \rho$ if $K, s \models \phi$ and $K, s \models \rho$
4. $K, s \models \phi \vee \rho$ if $K, s \models \phi$ or $K, s \models \rho$
5. $K, s \models \mathbf{A}\ \phi$ if $K, \pi \models \phi$ for all paths π with $\pi(0) = s$.
6. $K, s \models \mathbf{E}\ \phi$ if there exists a path π with $\pi(0) = s$ such that $K, \pi \models \phi$.
7. $K, \pi \models \phi$ if $K, \pi(0) \models \phi$
8. $K, \pi \models \mathbf{X}\ \phi$ if $K, \pi^1 \models \phi$
9. $K, \pi \models \phi\ \mathbf{U}\ \rho$ if there exists an integer $k \geq 0$ such that $K, \pi^k \models \rho$ and $K, \pi^j \models \phi$ for all $0 \geq j < k$.
10. $K, \pi \models \phi\ \mathbf{V}\ \rho$ if for every integer $k \geq 0$, $K, \pi^j \not\models \phi$ for all $0 \geq j < k$ implies $K, \pi^k \models \rho$.

If $K, s_0 \models \phi$ for all initial states $s_0 \in S_0$, then we write $K \models \phi$. We say that ϕ is contradicting the Kripke Structure if $K \not\models \phi$.

3 Constructing the World Model

We consider an information integration module that is connected with sensors. The sensors send events at certain points in time. The first step that has to be provided by the information integration module is to store the received events as corresponding atomic propositions into a Kripke Structure. For this purpose we assume that the sensors send an event E, or $\neg E$ at time t_E whenever a specific feature is detected or vanishes respectively. Hence, if E is received by the information integration model a corresponding atomic proposition is added to the current state. If $\neg E$ is received the corresponding atomic proposition is removed from the current state. A new state is generated whenever time t from one of the sensors advances. In this case the current state becomes the old one. A new current state is generated. This state comprises all atomic formulas from the old state. A directed edge between the old and the current state is added to the Kripke Structure. Formally, the algorithm for constructing the initial world model looks like follows:

1. **Initialization**
   (a) Initialize the Kripke Structure $K = (A, S_0, S, R, L)$. A is set to all atomic propositions that are used in the given background theory, $S_0 := S := \{s_0\}$, $L(s_0) := \emptyset$, $R := \emptyset$.
   (b) Set the current state s to s_0 and the current time t to 0.
2. **Event handling:** If an event or its negation $\psi \in \{E, \neg E\}$ is received from one of the sensors, do the following:
   (a) if $t_E > t$ then let the old state be the current one, i.e., $o := s$, generate a new current state s, add (o, s) to R, set $t := t_E$, and let $L(s) := L(o)$. Otherwise, do nothing.
   (b) If ψ is of the form E, then add the corresponding atomic proposition a_E to $L(s)$.
   (c) If ψ is of the form $\neg E$, then remove the corresponding atomic proposition a_E from $L(s)$.

Step 2 of the algorithm is executed as long as events come from the sensors or whenever the algorithm is stopped by an external event, i.e., the user of the MAS stops analyzing the meeting. The result of the algorithm is a Kripke Structure, i.e., the initial world model, which captures all observations. Because of the algorithm the initial world model has the same structure like a linked list. No alternative paths are represented and neither consistency nor completeness (wrt. the background knowledge) can be guaranteed.

4 Revising the World Model

The most important objective of the information integration module is to provide a consistent and complete world model with respect to the given background knowledge. For this purpose the initial world model has to be checked. In case of detected inconsistencies we are interested in changing the Kripke structure (i.e., the initial world model) such that the contradiction cannot be derived anymore. Moreover, in the particular application context we have to find a Kripke Structure which represents the model of the real world, e.g., a meeting, that is not in contradiction with all given CTL formulas representing the background theory. This problem is well known in literature and referred as belief revision problem. Buccafurri and colleagues [3] presented a solution to the belief revision problem of Kripke Structures. Their approach deals with changing the connections between the states of the Kripke Structure, focusing mainly on the repair of concurrent programs, and is based on other program repair approaches like Console and colleague's work [6].

Although the basic idea behind belief revision can be adapted to solve the belief revision problem for temporal knowledge bases in our area, there is one important difference which have to be taken into account. Changing the connections between the states of the Kripke Structure is not enough. There is one important reason that supports this observations. First, when starting to build the knowledge base from observations the resulting Kripke Structure is hardly

complete and consistent with respect to the given background knowledge. In order to ensure consistency, atomic propositions have to be removed from the states or new states representing alternative worlds and their corresponding connections have to be constructed. Moreover, for ensuring completeness new atomic propositions have to be added to the Kripke Structure. Hence, the label function of states, the set of states and their connectivity relation have to be updated.

In order to define formally the belief revision we first introduce the concept of update operators. An update operator either add an object to or remove an object from a part of the Kripke Structure. Depending on the part of the Kripke Structure, i.e., the states, the label function, or the state relation, we have different operators.

L-updates. The L-update operator $L^+(s,a)$ ($L^-(s,a)$) adds (removes) an atomic proposition a from the label function $L(s)$.
S-updates. The S-update operator $S^+(s)$ ($S^-(s)$) adds (removes) the state s from the set of states S of the Kripke Structure.
R-updates. The R-update operator $R^+(s,s')$ ($R^-(s,s')$) adds (removes) the tuple (s,s') from the set R.

$K|u$ denotes the application of an update operator u to a Kripke Structure K. Note that the update operators cannot be applied in an arbitrary order. For example trying to apply the L-update operator to a non-existing state will not change the Kripke structure. Moreover, after introducing a state it has to be connected to another state.

We now define an update U to a Kripke Structure $K = (A, S_0, S, R, L)$ as a sequence $\langle u_1, \ldots, u_n \rangle$. The result of applying U to K is inductively defined by:

1. $K|\langle u_1 \rangle = K|u_1$
2. $K|\langle u_1, u_2, \ldots u_n \rangle = (K|\langle u_1 \rangle)|\langle u_2, \ldots, u_n \rangle$

We define an update U as sound and complete iff the following rules are fulfilled:

- No single update operator of U tries to update a nonexistent part of the Kripke Structure during its application.
- After applying the U the following property must hold for the resulting Kripke Structure K'. For all states s in K' there must be a path from the initial state to s.
- The Kripke Structure must be consistent with the background theory T which is a CTL formula, i.e., $K, s_0 \models T$.

Of course, when given an initial model and a background theory, we are only interested in finding sound and complete updates. Unfortunately, searching for such an update is intractable because all possible combinations have to be checked. In order to speed up computation we suggest the use of heuristics that take care of the structure of the CTL formulas used in the background theory.

For example, consider the previously introduced formulas

$$AG\neg\,(Name(X,Y) \wedge Name(X,Z) \wedge Y \neq Z)\,and$$
$$AG\,(Name(X,Y) \rightarrow XName(X,Y))\,.$$

The first one is for checking consistency within a state whereas the latter is for ensuring completeness of atomic propositions along a path. In this case, the formular says that the name of the person remains the same in the next state. If we detect an inconsistency with respect to these formulas, we have to change the Kripke structure in order to distinguish two cases. Either the assigned name in the first occurrence is correct or the name in a later state is correct. A similar situation is depicted in Fig. 4(a). In this case the number of participants (*no_part*) is not allowed to change. A different correction is necessary for the situation in Fig. 4(b) where the consistency within one particular state is not ensured. In this example a detector delivers the observation that there is no noise (*no_noise*) but someone is speaking (*speaks*) which is obviously a case that cannot occur. The solution for this situation would be to introduce a new state to handle the inconsistency. Fig. 4(c) describes the situation where not all information is provided by the sensors. This information has to be restored by adding the necessary atomic propositions to the states. The computation of an update using the ideas described above would comprise the following steps. First, find out the reason for a detected inconsistency, e.g., a formula describing consistency requirements for a state is responsible for the inconsistency. Second, apply rules for adding new update operations. These rules have to correspond to the detected reason. For example, if a state inconsistency is detected, then it is necessary to introduce alternative states. These alternative states comprise

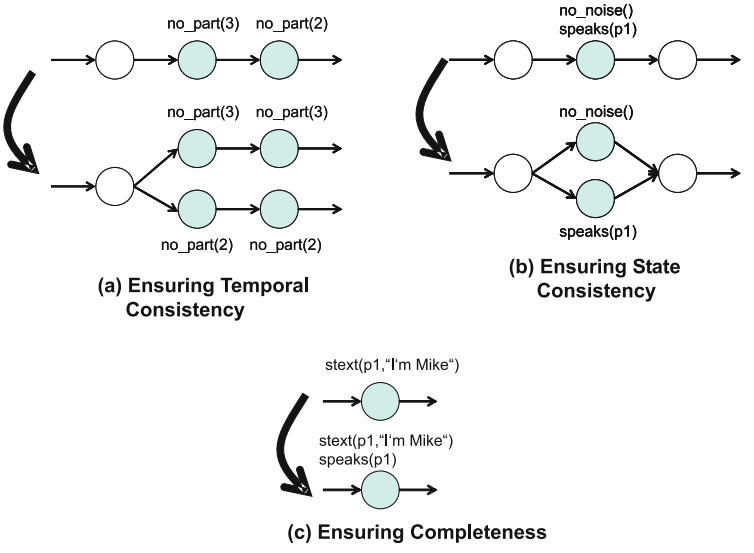

Fig. 4. Revising the knowledge base

all atomic propositions like the original state, except those propositions that are responsible for the conflict. For these propositions only one is allowed to occur in the original state and its alternative states. Third, use the modified Kripke Structure to check consistency again. If all consistencies are removed, the process stops and the update together with the finally created Kripke Structure is given back as result. Otherwise, a new reason has to be detected and so forth. Of course without further restrictions there is no guarantee that the proposed computation of the update halts. Future research has to deal with this issue.

5 Related Research

To our knowledge the use of Kripke Structures for representing world states together with CTL formulas for representing background knowledge is new. Previous research in this domain has mainly focused on using Kripke Structures and CTL for verification purposes, e.g., [4, 10]. The update or belief revision of Kripke Structures has been described before by Buccafurri et al. [3]. However, the authors introduce update only for the state relationship which seems to be enough for their purpose. Moreover, [3] is a very good source for reading about the relationship between the Kripke Structure revision and abductive reasoning for belief revision.

The work of Buccafurri et al. [3] makes use of the introduction of update operators like the work done by Console et al. [6]. In their paper and successor papers like [1, 2, 8] and most recently [9] the update operator is used in the context of debugging, namely fault detection, correction, repair. In their work there has been no necessity to handle temporal aspects which distinguishes their papers from ours.

There are of course also relationships to diagnosis approaches. Both to consistency-based diagnosis [11] or abductive diagnosis [5] since both approaches deal with removing inconsistencies by using either assumptions or operations. It would be of interest to show whether and how consistency-based diagnosis can be used to improve finding a complete and sound update for a knowledge base.

6 Conclusion

In this paper we introduced the use of Kripke Structures for representing temporal knowledge and CTL for formalizing background knowledge in domains where temporal aspects are important like the meeting domain where our examples come from. We further introduced an algorithm that allows for computing an initial knowledge base from event information provided by external sources like sensors. Because of the fact that such an obtained knowledge is neither correct nor complete with respect to a background theory, it is important to update the Kripke Structure. This is even more important in situations where the knowledge/observations come from unreliable sources like sensors. Therefore, we first introduced conceptually the concept of updating the initial knowledge which extends previous research. Second, we introduced heuristics for

computing the update which is based on the structure of the formulas within the background knowledge. A pattern matching approach would give us back the required changes. There are open problems which have to be solved. It is unclear whether the heuristics can be adapted and used for all formulas that occur in practice. Furthermore, requirements that allow the use of the heuristics have to be obtained. Moreover, a proof of concept study has to be carried out.

Acknowledgement

The project results have been developed in the MISTRAL project (http://www.mistral-project.at). MISTRAL is financed by the Austrian Research Promotion Agency (http://www.ffg.at) within the strategic objective FIT-IT under the project contract number 809264/9338.

References

1. G. W. Bond and B. Pagurek. A Critical Analysis of "Model-Based Diagnosis Meets Error Diagnosis in Logic Programs". Technical Report SCE-94-15, Carleton University, Dept. of Systems and Computer Engineering, Ottawa, Canada, 1994.
2. Gregory W. Bond. *Logic Programs for Consistency-Based Diagnosis*. PhD thesis, Carleton University, Faculty of Engineering, Ottawa, Canada, 1994.
3. Francesco Buccafurri, Thomas Eiter, Georg Gottlob, and Nicola Leone. Enhancing model checking in verification by ai techniques. *Artificial Intelligence*, 1999.
4. E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, April 1986.
5. Luca Console, Daniele Theseider Dupré, and Pietro Torasso. On the relationship between abduction and deduction. *Journal of Logic and Computation*, 1(5):661–690, 1991.
6. Luca Console, Gerhard Friedrich, and Daniele Theseider Dupré. Model-based diagnosis meets error diagnosis in logic programs. In *Proceedings* 13^{th} *International Joint Conf. on Artificial Intelligence*, pages 1494–1499, Chambery, August 1993.
7. E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. J. van Leeuwen, North-Holland, 1990.
8. Alexander Felfernig, Gerhard Friedrich, Dietmar Jannach, and Markus Stumptner. Consistency based diagnosis of configuration knowledge bases. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, Berlin, August 2000.
9. G. Friedrich and K. Shchekotykhin. Diagnosis of description logic knowledge bases. In *Working Notes of the IJCAI-05 Workshop on Model-Based Systems*, Edinburgh, Scotland, August 2005.
10. Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993. ISBN 0-7923-9380-5.
11. Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.