A Belief Update System Using an Event Model for Location of People in a Smart Home

Marie Bernert and Fano Ramparany^(✉)

Orange Labs, 28 Chemin du Vieux Chêne, 38240 Meylan, France
fano.ramparany@orange.com

Abstract. Artificial Intelligence applications often require to maintain a knowledge base about the observed environment. In particular, when the current knowledge is inconsistent with new information, it has to be updated. Such inconsistency can be due to erroneous assumptions or to changes in the environment. Here we considered the second case, and develop a knowledge update algorithm based on event logic that takes into account constraints according to which the environment can evolve. These constraints take the form of events that modify the environment in a well-defined manner. The belief update triggered by a new observation is thus explained by a sequence of events. We then apply this algorithm to the problem of locating people in a smart home and show that taking into account past information and move's constraints improves location inference.

Keywords: Belief revision · Event logic · Semantic reasoning · Smart home · IoT

1 Introduction

A smart home should provide adapted services to its inhabitants. Indeed, the users' needs strongly depend on who is present in the house, where are the people located, what they are doing, at which time of the day and which day of the week, and so on. It is thus crucial to infer this context from the data provided by the house equipment. For example, concerning the “where” part of the context, the precise location of an occupant in the house can be used, among other, to chose a device to communicate with this occupant, or to suggest activities linked to this location. However, sensors' location information are often sparse and imprecise, due to the cost of equipping a house with numerous devices, and the rejection of too intrusive devices such as cameras. As an example of an easily available but vague information, a motion detector provides the information that at least one person is present in a room. Similarly, a smartphone WiFi connection provides the information that its owner is near or in the house. In spite of this vagueness, useful information can be inferred by tracking location information over time and taking into account the house topology. More generally, in many cases a knowledge about an environment is inferred from only sparse information. However, knowing the evolution constraints of the environment and accumulating information over time can lead to a substantial knowledge about the environment, as we do in our everyday life. Our goal is to implement an algorithm that takes location information

from sensors of a smart home, and infer people location from this information, taking into account constraints on moves. More generally, we propose an algorithm able to revise knowledge taking into account well defined evolution constraints.

2 Use Case Example

In this section we present a use case scenario, defined as a main test case to design and test our location algorithm. In this scenario, we consider a simple house composed of four rooms and inhabited by two people: Alice and Bob. The four rooms are the entrance connected with the outside, the kitchen connected to the entrance, the living-room also connected to the entrance and the bedroom connected to the living-room (see Fig. 1). The home is equipped with some sensors that can give us information about people location:

- The entrance and living room are both equipped with a presence detector informing us whether some people are present in the room or not.
- The kitchen is equipped with a smart fridge informing us if someone is opening the fridge's door and is thus present in the kitchen.
- Bob is carrying a smart device informing us whether Bob is inside or outside the house.

Given the house topology and its sensor equipment, we consider the following seven-step scenario:

- step 1: Alice and Bob are both outdoor.
- step 2: Alice enters the house and is now in the entrance
- step 3: Alice goes in the living room.
- step 4: Bob then enters the house and is now in the entrance
- step 5: Bob goes in the kitchen.
- step 6: Alice then goes in the bedroom.
- step 7: Bob opens the fridge, notifying a presence in the kitchen.

When considering only the last step of the scenario, the location devices inform us that somebody is in the kitchen, nobody is in the entrance nor the living room and Bob is somewhere in the house. We infer from this information that Alice or Bob is in the kitchen, Bob is in the kitchen or the bedroom, and Alice is in the kitchen or in the bedroom or outdoor. However, when considering all the sensor information from the beginning of the scenario together with the room adjacency constraints, one can easily infer that Alice is in the bedroom and Bob is in the kitchen. This simple example shows that it is possible to infer much more information by taking into account the house's topology and past information. This use case can be used to discriminate an algorithm that uses such a strategy from one that does not.

3 Related Work

3.1 Logical Formalism

To address our problem, we need a logical formalism to deal about events and evolving facts. Many logical systems have been defined for this purpose. Here we present some of them.

Fig. 1. Example of a smart home, equipped with simple location devices

Dynamic Logic. Dynamic logic was originally developed to reason about computer program, in particular to verify their correctness. Hoare’s logic constitute a well known example of programming logic [6]. It was later realized that such logic could be used for other applications and it was then generalized to dynamic logic. Meyer gives a review of different dynamic logic applications [7]. In the context of our problem, we are interested by dynamic logic used as a logic of action.

Dynamic logic of action is build on a logical language $\mathcal{L}_{DL}$ and an action language $\mathcal{L}_{ACT}$. $\mathcal{L}_{DL}$ includes a set of propositional atoms P and is closed under the usual syntactic rules. $\mathcal{L}_{ACT}$ includes a set of atomic actions A and is closed under rules such as sequential composition of action $(\alpha; \beta)$, choice of action $(\alpha + \beta)$, arbitrary finite repetition of action (α^*) , with $\alpha, \beta \in A$. In addition to the usual syntactic rules, $\mathcal{L}_{DL}$ is closed under the following rule: if $\phi \in \mathcal{L}_{DL}$ and $\alpha \in \mathcal{L}_{ACT}$ then $[\alpha]\phi$ and $\langle\alpha\rangle\phi$ are in $\mathcal{L}_{DL}$.

An interpretation for $\mathcal{L}_{DL}$ is a structure $\mathcal{M}$ of the form (S, π, r) where S is a non-empty set of states, $\pi : S \times P \rightarrow \text{BOOL}$ is a truth assignment function that associates a truth value to each couple of state and atomic proposition, and $r : \mathcal{L} \rightarrow \mathcal{P}(S \times S)$ a function that associates a state transition relation to each action. Given an interpretation $\mathcal{M} = (S, \pi, r)$ and a state $s \in S$, the truth value of a formula $\phi \in \mathcal{L}_{DL}$ is defined by:

- $\mathcal{M}, s \models p$ iff $\pi(s, p) = \text{true}$, for $p \in P$.
- $\mathcal{M}, s \models \neg\phi$ iff not $\mathcal{M}, s \models \phi$.
- $\mathcal{M}, s \models \phi \wedge \psi$ iff $\mathcal{M}, s \models \phi$ and $\mathcal{M}, s \models \psi$.
- $\mathcal{M}, s \models \phi \vee \psi$ iff $\mathcal{M}, s \models \phi$ or $\mathcal{M}, s \models \psi$.
- $\mathcal{M}, s \models \phi \rightarrow \psi$ iff $\mathcal{M}, s \models \phi$ implies $\mathcal{M}, s \models \psi$.
- $\mathcal{M}, s \models \phi \leftrightarrow \psi$ iff $\mathcal{M}, s \models \phi$ bi-implies $\mathcal{M}, s \models \psi$.
- $\mathcal{M}, s \models [\alpha]\phi$ iff for all s' such that $(s, s') \in r(\alpha)$ we have $\mathcal{M}, s' \models \phi$.
- $\mathcal{M}, s \models \langle\alpha\rangle\phi$ iff for some s' such that $(s, s') \in r(\alpha)$ we have $\mathcal{M}, s' \models \phi$.

Dynamic logic of action allows to write formulas such as $\phi \rightarrow [\alpha]\psi$, meaning that if ϕ is true, then executing action α leads to ψ being true. Is is thus possible to describe the result of an action. However this formalism can not deal with an explicit timeline, and it is not possible, for example, to assert that an action occurred at a specific time point, or that a proposition is true during a given time interval.

Event Logic. In [3], Allen developed a temporal logic, based on predicates logic, to reason about actions. This logical formalism involves properties, events, and time intervals. Allen defines a set of thirteen mutually exclusive primitive relation between intervals (see Fig. 2), originally developed in [2]. This set is $\mathcal{R}_{Allen} = \{=, <, >, m, o, d, s, f, mi, oi, di, si, fi\}$ where $=, <, >, m, o, d, s, f$ respectively stands for “equals”, “before”, “after”, “meets”, “overlap”, “during”, “start”, “finish”, and each xi is the inverse of x . According to these relations, the predicates $STARTS(I, J)$, $FINISHES(I, J)$, $BEFORE(I, J)$, $OVERLAP(I, J)$, $MEETS(I, J)$ as well as $EQUAL(I, J)$ are defined, for times intervals I and J . Two other predicates are defined. The first is $HOLDS(p, I)$, where p is a property and I a time interval, meaning that property p is true during the interval I . The second predicate is $OCCURS(e, I)$, where e is an event and I a time interval, meaning that the event e occurs during the interval I , in other words e begins at the beginning of I and ends at the end of I . Whereas a property p holding during an interval I also holds for all sub-intervals of I , an event e can not be split, and its occurrence coincide exactly with the interval I . Allen also defines some other predicates about processes, causality and actions, which we will not detail here.

Fig. 2. Allen's primitive relations between intervals

Following this idea of reasoning about event occurrences, Siskind developed a logic known as event logic [8]. A language $\mathcal{L}_{EL}$ of event-logic expressions is defined as follow. A finite set O of constant symbols and a finite set E of primitive event-type symbols are given. An atomic event-logic expression is defined as a primitive event-type symbol of arity n applied to a sequence of n constants. Finally, an event-logic expression is either an atomic event-logic expression or a compound expression: $\neg\phi$, $\phi \vee \psi$, $\phi \wedge_R \psi$, $\Diamond_R \phi$, with $R \subseteq \mathcal{R}_{Allen}$, and ϕ and ψ event-logic expressions. An event-occurrence formula has the form $\phi@I$ where $\phi \in \mathcal{L}_{EL}$ and I is a time interval. An interpretation $\mathcal{M}$ is a function that associate each primitive event-type symbol of arity n to a subset of $\mathcal{I} \times O^n$, where $\mathcal{I}$ is the set of all time intervals. The truth value of the formula $\phi@I$ relatively an interpretation $\mathcal{M}$ is define by:

- $\mathcal{M} \models e(o_1, \dots, o_n)@I$ iff $(I, o_1, \dots, o_n) \in \mathcal{M}(e)$, for $e \in E$.
- $\mathcal{M} \models (\neg\phi)@I$ iff not $\mathcal{M} \models \phi@I$.
- $\mathcal{M} \models (\phi \vee \psi)@I$ iff $\mathcal{M} \models \phi@I$ or $\mathcal{M} \models \psi@I$.
- $\mathcal{M} \models (\phi \wedge_R \psi)@I$ iff there exists time intervals J and K such that I is the smallest super-interval of both J and K , JrK for some $r \in R$, $\mathcal{M} \models \phi@J$ and $\mathcal{M} \models \psi@K$.
- $\mathcal{M} \models (\diamond_R \phi)@I$ iff there exist some time interval J such that JrI for some $r \in R$, and $\mathcal{M} \models \phi@J$.

It is possible to define a primitive event-type, denoted $\bar{\phi}$, from a predicate ϕ . The event occurrence $\bar{\phi}@I$ is true if the predicate ϕ is true at each point of the interval I . This allows to unify the concepts of events and properties defined by Allen in [3]. As an example let's assume that we are given a set of persons, a set of rooms, and a predicate symbol *IsIn*. The predicate *IsIn*(p, r), for a person p and a room r , is true at time t if p is present in r at time t . We can define the compound event-type expression *Move*(p, r_1, r_2), for a person p and rooms r_1 and r_2 as follow: $\text{Move}(p, r_1, r_2) = \Diamond_{\{m\}} \overline{\text{IsIn}}(p, r_1) \wedge_{\{=\}} \Diamond_{\{mi\}} \overline{\text{IsIn}}(p, r_1) \wedge_{\{=\}} \neg \overline{\text{IsIn}}(p, r_1) \wedge_{\{=\}} \overline{\text{IsIn}}(p, r_2)$. This states that a person p is moving from room r_1 to room r_2 during interval I iff p is in r_1 just before I , in r_2 just after I and that p is not in r_1 nor in r_2 during I .

3.2 AGM Model

The AGM model was developed by Alchourrón, Gärdenfors and Makinson as a framework for belief revision [1]. Its main goal is to define good properties of a revision operation on a belief set. A good introduction to the AGM model is given by Fermé [5]. Here we detail the main features of the AGM Model.

A belief set, or theory, K is a subset of a logical language $\mathcal{L}$ that is closed under logical consequence. Denoting Cn the consequence operation, we thus have $K = Cn(K)$. Given a belief set K a statement x , x is either believed if $x \in K$, disbelieved if $\neg x \in K$, or unsettled otherwise. The purpose of belief revision is to add or retract statements from a belief set. The AGM model define the possible revision operations on a belief set and give some postulates these operations should satisfy. Given a belief set K and a statement x , three operations are possible:

- expansion, denoted $K + x$, which changes the state of x from unsettled to believed
- contraction, denoted $K - x$, which changes the state of x from believed to unsettled
- revision, denoted $K * x$, which changes the state of x from disbelieved to believed

Expansion can be easily defined as $K + x = Cn(K \cup \{x\})$, effectively adding x to K without removing or adding information unnecessarily. Moreover, as in this case $\neg x \notin K$, if $Cn(x)$ is consistent the result is also consistent. When $\neg x \in K$, adding x to K is a revision operation. It is necessary to first remove $\neg x$ from K before adding x . The revision operation can be defined using the contraction operation through Levi identity: $K * x = Cn((K - \neg x) \cup \{x\})$. If the contraction is consistent and successful, then the revision operation is also consistent and successful.

The key of the problem is thus the contraction operation. The AGM model defines 6 main postulates a contraction operation should satisfy:

- closure: $K - x = Cn(K - x)$.
- inclusion: $K - x \subseteq K$.
- vacuity: if $x \notin K$ then $K - x = K$.
- success: if $x \notin Cn(\emptyset)$ then $x \notin K - x$.
- preservation: if $Cn(x) = Cn(y)$ then $K - x = K - y$.
- recovery: $K \subseteq Cn((K - x) \cup \{x\})$.

As a tool to define contraction, we denote $K \perp x$ the set of all maximal subset of K that does not imply x . A first naive approach to define contraction, called maxichoice contraction, is to chose $K - x$ to be one element of $K \perp x$. The maxichoice contraction has some disconcerting properties. In particular, when defining revision through the Levi identity, $K * x$ is always complete, which means that no statement is unsettled. Thus, the belief set generated by maxichoice contraction and revision might be considered “too big”. A second approach, called meet contraction, is to define $K - x$ as the intersection of all elements of $K \perp x$. In this case, on the contrary, the result might be considered “too small”. Indeed we have $K - x = K \cap Cn(x)$ and $K * x = Cn(x)$. In between, contraction can be defined as a partial meet contraction, which consist in selecting the most important elements of $K \perp x$. Let γ be a selection function such that $\gamma(K \perp x)$ is a non-empty subset of $K \perp x$. Partial meet contraction is defined as $K - x = \bigcap \gamma(K \perp x)$ and partial meet revision is defined through the Levi identity. Maxichoice contraction and meet contraction are extreme cases of partial meet contraction, where γ selects respectively one element or all elements of $K \perp x$. It can be shown that a contraction operation satisfies the 6 postulates if and only if it is a partial meet contraction. There is thus no general way to define contraction (and revision). Contraction requires to make some choice about the interesting beliefs to be preserved.

The AGM is a general framework for belief revision, that gives properties a revision operation should satisfy. However it does not detail practical implementation of these operation. In particular the contraction operation is not trivial to define.

3.3 Truth Maintenance Systems

Truth maintenance systems (TMS) were introduced by Doyle in [4]. As for most belief revision systems, Doyle’s TMS tackles the problem of revising a belief set when a new information brings a contradiction. The two main principles of the TMS is to use a non-monotonic logic, where some facts are believed unless proved false, and to keep track of reasons why a fact is believed.

The TMS works in duality with a problem solver, which provides statements and justifications for these statements. The goal of the TMS is to decide which statement should be believed or not depending on their justifications. Within the TMS, statements are represented by node that are said to be “in” if the statement is believed or “out” otherwise. One node is marked as a contradiction and should not be “in”. A justification for a node consists in two parts: a in-list and an out-list. A justification makes a node “in” iff all nodes in the in-list are “in” and all nodes in the out-list are “out”. The out-list constitutes the non-monotonic part of the TMS. For example, in natural language, a justification for “Titi can fly” can be: “If Titi is a bird, Titi can fly, unless it is a penguin”. In the TMS formalism, the node “Titi can fly” has a justification with the

in-list “Titi is a bird” and the out-list “Titi is a penguin”. There are particular of node in the TMS, called assumptions, which are nodes justified by an out-list containing their negation. An update of the TMS is triggered when a new justification is added. When the contradiction node becomes “in” after an update, a backtracking procedure is called to make the contradiction “out” again. This is done by finding the assumptions that justify (possibly indirectly) the contradiction and making one of these assumption “out” by adding a justification.

The TMS approach explains a contradiction by the fact that some assumptions were made that are not true. The contradiction is solved by disbelieving these assumptions. In our case, a contradiction can arise if a fact that was previously true becomes false because the environment is evolving. This difference makes the TMS not suitable for our problem.

4 Our Contribution

4.1 Algorithm Overview

Our algorithm assumes that we are provided information from sensors, that holds during a time interval. The time line is divided into time intervals, each time interval corresponding to a set of observations that holds during the entire interval. It is also possible to have intervals during which no information is given. In addition to these observations about the environment we are given a set of events that can make the environment evolve. We assume that these events modify our knowledge in a relatively simple way, such that, given a belief set holding before the event, we know what new belief set holds after an event occurs. The goal is to infer facts about the current environment from the consecutive observations and the possible event sequences explaining these observations.

The principle of the algorithm is to explore all possibilities of event sequences compatible with the past and current observations. Possibilities are explored by examining the consequences of adding an event to a sequence that has already been considered. The added event should be compatible with the current observations and what had already been inferred from the previous sequence hypothesis. Each time the observations change, new possibilities can be explored. Once all possibilities have been explored, we can infer that a fact about the environment is true if it is true considering every event sequence hypothesis, or possible if it is possible for at least one sequence hypothesis.

4.2 Logical Formalism

For the purpose of our problem, we found it practical to reason about continuous time and punctual events (events occurring at a precise time point). Indeed, for simplicity we assume that properties, such as the room position of a person in the house, are discrete and always well defined. As a consequence, changes on properties, such as the move of a person from one room to an adjacent room, are punctual events. Event logic provides useful operators to reason about event occurrences over time. However, for more flexibility, we chose to define a logic based on predicate logic as it was done in [3].

The main idea is to take a classical logic, later called the base logic, and augment it with time and events to construct a dynamic logic. We simply assume that the base logic contains the conjunction and the disjunction. Formulas from the base logic will be later called properties. A finite set E of punctual event symbols is given. A interpretation for our punctual event logic consists in two main elements:

- A transition model: each event symbol is associated with a transition function, which itself associates each base logic interpretation to another base logic interpretation. In other words the transition model defines how each event modifies a base logic interpretation.
- A sequence of event occurrence: An initial base logic interpretation and a sequence of events symbols associated with time points is given. Taking into account the transition model, a sequence of base interpretation associated with consecutive time intervals can be inferred, defining which properties are true at each time point.

We define two main predicates to write event logic formulas:

- $Occurs(e, t)$, with e an event symbol and t a time point, meaning that the event e occurs at time t , according to the event sequence model.
- $Holds(\phi, I)$, with ϕ a property and I a time interval, meaning that the property ϕ is true during the interval I according to the possible base logic models during this time interval.

We also define some other useful derived predicates:

- $Idle(I)$, meaning that no event occurs during the time interval I .
- $OccursSeq(s, I)$, meaning that the sequence of event $s = (e_1, \dots, e_n)$ (and no other event) occurs during the time interval I
- $Holds_R(\phi, I)$, meaning that there exist a time interval J and an Allen relation $r \in R$, such that JrI and $Holds(\phi, J)$
- $Occurs_R(e, I)$ meaning that there exist a time interval I and an Allen relation $r \in R$, such that $Ir\{t\}$ and $Occurs(e, t)$
- $OccursSeq_R(s, I)$ meaning that there exist a time interval J and an Allen relation $r \in R$, such that JrI and $OccursSeq(s, J)$

This formalism gives us a framework to design our algorithm.

4.3 Transition Graph Structure

Let assume that we made a series of observations $O_0, \dots, O_n$ during consecutive intervals $I_0, \dots, I_n$. Observations $O_0, \dots, O_n$ are sets of properties. We can thus write for each k : $Holds(O_k, I_k)$. The intervals I_k are called observation intervals. Note that the observation intervals do not necessarily coincide with the intervals during which the base logic model does not change. The observations can change without an event occurring and an event can occur without inducing a change in the observations. We assume that the transition model is known. However the event occurrence succession is unknown. Our goal is to infer properties given the observations and the transition model.

Let focus on one particular time point t in an interval I_k . The main idea of our algorithm is to make hypotheses about the event sequences that occurred from the starting time point t_0 until t . For this purpose, we associate each event sequence s to its transition function T_s , which is the composition of the transition functions associated to each of its events. Given a transition function T , we denote $Seq(T)$ the set of event sequences s such that $T_s = T$. Given a time point t , an observation interval I_k and a transition function T , we consider, as an event sequence hypothesis, the formula, denoted $N_T^k(t)$, stating that t is in I_k and that the event sequence between t_0 and t belong to $Seq(T)$:

$$N_T^k(t) = t \in I_k \wedge \exists s \in Seq(T), OccursSeq(s, [t_0; t]) \quad (1)$$

The formula $N_T^k(t)$, will be later called a belief node, as we will build a graph structure on these hypotheses. Let first notice that, for k given, the disjunction of the $N_T^k(t)$ for all transition function T is simply the statement that t belongs to I_k . For a given k , the observation interval I_k can be thus associated with the set of belief nodes $N_T^k(t)$ with T ranging over all possible transition function.

Let us now build a graph structure on belief nodes. For this purpose, we build an equivalent formula for $N_T^k(t)$ using predecessor belief nodes. Given a transition function T we consider $Pred(T)$ the set of couple (T', e) , with T' a transition function and e an event symbol, such that $T = T_e \circ T'$. The hypothesis $N_T^k(t)$ is true if and only if one of the following hypothesis is true:

- for one $(T', e) \in Pred(T)$, there exists $t' < t$ such that $N_{T'}^k(t')$ and e is the only event occurring between t' and t .
- $k > 1$ and for one $(T', e) \in Pred(T)$, there exists $t' < t$ such that $N_{T'}^{k-1}(t')$, e occur at the time point between I_{k-1} and I_k and e is the only event occurring between t' and t .
- $k > 1$ and there exists $t' < t$ such that $N_{T'}^{k-1}(t')$, and no event occurs during t' and t .
- if $k = 0$ and T is the identity, we also need to consider the hypothesis that no event occurs between t_0 and t .

Thus, each belief node can be written as a disjunction of hypotheses involving other belief nodes, which are predecessor belief nodes through different events. A predecessor belong to the same observation interval when the last event occurred in the this interval, or to the previous observation interval when the last event occurred at the time point between the two intervals (in this case the transition can correspond to no event).

The belief nodes can thus be organized into a graph, which we call transition graph, where vertices are belief nodes and edges correspond to events (see Fig. 3). The edges have two different types: internal edges, linking nodes corresponding to the same observation interval, and external edges, linking nodes from two consecutive observation intervals. We thus label the edges with transition symbols constructed from the event symbols and taking into account the internal or external nature of the edge. We denote this set of transition symbol $E_{tr} = E_{in} \cup E_{ex}$ with $E_{in} = \{(e, in), e \in E\}$ the set of internal transition symbols and $\{(e, ex), e \in E \cup \{idle\}\}$ the set of external transition symbols. The successor of a node N_T^k through an edge labeled by $e \in E_{tr}$ can be easily computed as $succ(N_T^k, e) = N_{T_e \circ T}^k$ if $e \in E_{in}$ and $succ(N_T^k, e) = N_{T_e \circ T}^{k+1}$ if $e \in E_{ex}$.

A walk in the transition graph starting from the initial node N_{Id}^0 gives a sequence of events for which we know the position relatively to the observation intervals. The definition of the hypothesis $N_T^k(t)$ can be refined by stating that there exist some walk w from N_{Id}^0 to $N_T^k(t)$ such that the events occurring between t_0 and t correspond to the event sequence described by w with the correct position in the observation intervals.

Fig. 3. Example of transition graph structure

4.4 Nodes' Belief Sets

A belief set is a set of formulas on a logical language, closed upon logical consequence. For convenience we will use belief set within logical formulas. In such cases, the belief set can be seen as the conjunction of all its elements. Similarly we sometimes define the value of a belief set through a logical formula, implicitly meaning that the belief set is the set of consequences from this formula. A belief set can also be seen as a set of interpretations, corresponding to the interpretations upon which all its formulas are true. From this point of view the conjunction (resp. disjunction) of two belief sets is the intersection (resp. union) of the corresponding sets of interpretations. In our logical formalism, a transition function can be applied to a base logic belief set, using the correspondence with set of interpretations. Notice that any transition function preserves the conjunction and the disjunction, and is monotonic relatively to the implication.

To each belief node N_T^k we associate a base logic belief set B_T^k containing all properties that can be inferred at a time point t upon the $N_T^k(t)$ hypothesis and given the past observations. In other words, we want to find B_T^k such that, for $t \in I_k$:

$$N_T^k(t) \wedge \bigwedge_{k'=0}^k Holds(O_{k'}, I_{k'}) \rightarrow Holds_{\{m\}}(B_T^k, \{t\}) \quad (2)$$

Let w be a walk from the initial node N_{Id}^0 to a node N_T^k . We associate w to a belief set $B(w)$ such that if the sequence of events described by w occurred between t_0 and the current time point t , and the taking into account the past observations $O_0, \dots, O_k$, then we have $Hold_{\{f\}}(B(w), \{t\})$. This belief set can be defined the following way:

$$B(w) = \begin{cases} O_0 & \text{if } p \text{ is empty} \\ T_e(B(w')) \wedge O_k & \text{with } w = (w', e) \text{ and } k = n_{ex}(w) \end{cases} \quad (3)$$

where $n_{ex}(w)$ is the number of external event transitions in w . Using $Hold_{\{m\}}(B, t) \wedge Occurs(e, t) \rightarrow Hold_{\{mi\}}(T_e(B), t)$ as well as $Hold_{\{s\}}(B, I) \wedge OccursSeq(s, I) \rightarrow Hold_{\{f\}}(T_s(B), I)$, it can be shown that $B(w)$ defined this way satisfies the desired property.

As the hypothesis $N_T^k(t)$ states that there exist some walk w from N_{Id}^0 to $N_T^k(t)$ such that the corresponding sequence occurred, the belief set B_T^k can be defined as follow to satisfy Eq. 2:

$$B_T^k = \bigvee_{w \in walk(N_{Id}^0, N_T^k)} B(w) \quad (4)$$

where $walk(N_{Id}^0, N_T^k)$ is the set of walk from the initial node to N_T^k in the transition graph.

The goal of the algorithm is to compute the belief set B_T^k recursively.

4.5 Building the Graph

The goal of the algorithm is to build the transition graph and compute the nodes' belief sets recursively to match Eq. 4 so that Eq. 2 is satisfied for all belief nodes. As an input, the algorithm is provided, one after the other, the observations associated to each observation interval. Each time the observation associated to the next interval is received, the algorithm update the graph to compute the nodes associated to this interval.

The following notations are used to describe the algorithm:

- An observation interval I is identified to its set of associated nodes, so we can write $N \in I$ if the node N is associated to I .
- For an observation interval I , $next(I)$ denotes the following consecutive interval: $next(I_k) = I_{k+1}$.
- For a node N , $Obs(N)$ is the set of observation associated to the interval it belongs to.
- For a node N , $B(N)$ is the belief set associated to N .

All nodes' belief sets are initialized to be inconsistent. When the observations associated to the first observation interval is given, the *UpdateInitialInterval* function is called (see algorithm 1). Then, each time the observations associated to the next observation interval is received, the *UpdateNextInterval* function is called (see algorithm 2). These two function ensure that the $I_{current}$ variable refer to the last observation interval for which information has been received, and that the belief sets of nodes associated to this interval (and all previous intervals) are correctly computed. These two

function both call the recursive function *UpdateNode* (see algorithm 3), which performs a deep first exploration of the graph, updating the node's belief set when necessary.

Each call of the recursive function *UpdateNode* corresponds to a walk in the graph. We define the set W of explored walks as the maximal set of walks such that for each node N , the associated belief set is the disjunction of all $B(w)$ with $w \in \text{walk}(N_{Id}^0, N) \cap W$. For short, we note $W_N = \text{walk}(N_{Id}^0, N) \cap W$. The algorithm is correct if at the end of all recursive calls, W contains all walks from the initial node to the current observation interval. During the algorithm execution, the following property on W is maintained: if W contains a walk w which is not in the call stack, W also contains all walks starting with w . In particular, when the algorithm terminates, the call stack is empty and, as W contains all walks in the previous interval (or the empty walk), W also contains all walks in the current interval. To maintain this property, the *UpdateNode* function ensures that if $W_{N_{pred}}$ contains a walk w when it is called, then at the end of the execution, W contains all walks beginning with (w, e) . In the recursive case, the node's belief set is updated so that W_N contains (w, e) . As the function is then called recursively for all event transitions e' , W should contain all walks starting with (w, e, e') for all e' , and thus all walks starting with (w, e) . In the base case, the belief set update has no effect, which means (w, e) is already in W_N while not in the call stack, and thus all walks beginning with (w, e) are already in W .

A key result for the termination of the algorithm is that cycles in a walk w do not impact the computation of $B(w)$. Indeed if $w = w'w_c$ with w_c a cycle, it can be shown, using the monotonicity of transition functions, that $B(w) \rightarrow B(w')$. When the *UpdateNode* function is called for a walk $w = w'w_c$ with w_c a cycle, the corresponding node N has already been updated so that w' is in W_N . Assuming that w_c is the only cycle in w , the condition on the structure of W ensure for all walks w'' in $W_{N_{pred}}$, (w'', e) is already in W_N , except for w . As $B(w) \rightarrow B(w')$, w is also already in W_N . The update has thus no effect on the belief set and the function returns. As a consequence walks with cycles are never effectively explored, which ensure that the algorithm terminates, as long as the number of consistent nodes is finite.

Algorithm 1. *UpdateInitialInterval()*

```

 $I_{current} := I_0$ 
 $B(N_{Id}^0) := O_0$ 
for  $e \in E_{in}$  do
  UpdateNode( $\text{succ}(N_{Id}^0), N_{Id}^0, e$ )
end for

```

Algorithm 2. *UpdateNextInterval()*

```

for  $N \in I_{current}$  do
  for  $e \in E_{ext}$  do
     $UpdateNode(succ(N, e), N, e)$ 
  end for
end for
 $I_{current} := next(I_{current})$ 

```

Algorithm 3. *UpdateNode(N, N_{pred}, e)*

```

 $B_{old} := B(N)$ 
 $B(N) := B(N) \vee (T_e(B(N_{pred})) \wedge Obs(N))$ 
if  $B(N) \neq B_{old}$  then
  for  $e \in E_{in}$  do
     $UpdateNode(succ(N, e), N, e)$ 
  end for
end if

```

4.6 Querying the Graph

Once the transition graph is constructed, we want to know, given a time point t , which properties are true at this time point. By construction, during an observation interval, the disjunction of all its belief nodes holds. Thus, a property is true at a time point within the observation interval if it is true in all belief nodes. Additionally, a property is possible (i.e. not false) if it is possible in at least one belief node. One can also get interested in what happen at the beginning (resp. at the end) of the observation interval, by looking only at the belief nodes that have consistent predecessors (resp. successors) in the previous (resp. next) interval. For example, a property is true at the beginning of the observation interval if it is true in all nodes that have a consistent predecessor in the previous interval. Knowing which properties are true at the beginning, during or at the end of each observation interval, we can infer if a formula of the form $Hold_{s_R}(\phi, I)$ is true, false or unknown according to current knowledge. Moreover possible event sequences from t_0 to a time point t in I_k correspond to walks in the graph from the initial node N_{Id}^0 to a node associated to I_k , going through only consistent nodes.

5 Application to the Location Problem

We will now apply this algorithm to the home location problem. Here we assume that the house topology is known, and that a set devices provides two type of location information over time: information about the number of people present in one room, and information about the location of a specific person. We also assume that only known people are present in the house. We thus have a set of person P , a set of rooms R and an adjacency relation $Adj \subseteq R \times R$. The property language is build with one predicate: $IsIn(p, r)$ with $p \in P$ and $r \in R$. An event is a person moving from one room to an adjacent room. The set of event symbols is defined as: $E = \{Move(p, r_1, r_2), p \in P, (r_1, r_2) \in Adj\}$. A transition function corresponds to a

subset of person moving each from one room to another room. For convenience, as they lead to similar belief sets, we chose to group in the same belief node all transition functions for which people arrive in the same position, not taking into account their initial position. We consider a belief set as a disjunction of house states, where a house state is the conjunction of predicates of the form $IsIn(p, r)$, for all $p \in P$. A house state thus describes the position of all people in the house, and can be seen as an interpretation for the base logic. Here for convenience, we use a different language for observations. An observation can be whether the predicate $Count(r, N)$, with $r \in R$ and $N \subseteq \mathbb{N}$, meaning that the number of persons in room r is in N , or the predicate $Located(p, R')$, with $p \in P$ and $R' \subseteq R$, meaning that the person p is in one of the room in R' . Adding an observation to a belief set can be simply done by removing the incompatible house states from the disjunction.

We applied this algorithm to the use case described in Sect. 2. We have $P = \{A, B\}$ for Alice and Bob, and $R = \{o, e, k, l, b\}$ for outdoor, entrance, kitchen, living-room and bedroom. The sensors deliver information about Bob's location, and the number of person in the entrance, the living-room and the kitchen. We denote $out = \{o\}$,

Table 1. Results of the algorithm applied to Alice and Bob use case

Node	Belief
Interval 1: $Located(B, out), Count(e, zero), Count(l, zero)$	
$\{\}$	$(IsIn(A, o) \wedge IsIn(B, o))$ $\vee (IsIn(A, b) \wedge IsIn(B, o))$ $\vee (IsIn(A, k) \wedge IsIn(B, o))$
Interval 2: $Located(B, out), Count(e, some), Count(l, zero)$	
Move(A, *, e)	$IsIn(A, e) \wedge IsIn(B, o)$
Interval 3: $Located(B, out), Count(e, zero), Count(l, some)$	
Move(A, *, l)	$IsIn(A, l) \wedge IsIn(B, o)$
Interval 4: $Located(B, home), Count(e, some), Count(l, some)$	
Move(A, *, l), Move(B, *, e)	$IsIn(A, l) \wedge IsIn(B, e)$
Interval 5: $Located(B, home), Count(e, zero), Count(l, some)$	
Move(A, *, b), Move(B, *, l)	$IsIn(A, b) \wedge IsIn(B, l)$
Move(A, *, l), Move(B, *, b)	$IsIn(A, l) \wedge IsIn(B, b)$
Move(A, *, l), Move(B, *, l)	$IsIn(A, l) \wedge IsIn(B, l)$
Move(A, *, l), Move(B, *, k)	$IsIn(A, l) \wedge IsIn(B, k)$
Interval 6: $Located(B, home), Count(e, zero), Count(l, zero)$	
Move(A, *, b), Move(B, *, k)	$IsIn(A, b) \wedge IsIn(B, k)$
Move(A, *, b), Move(B, *, b)	$IsIn(A, b) \wedge IsIn(B, b)$
Interval 7: $Located(B, home), Count(e, zero), Count(l, zero), Count(k, some)$	
Move(A, *, b), Move(B, *, k)	$IsIn(A, b) \wedge IsIn(B, k)$

$home = \{e, k, l, b\}$, $zero = \{0\}$ and $some = \mathbb{N}^*$. The scenario is composed of seven steps, corresponding to observation intervals. The transition graph resulting from the algorithm is described in Table 1. Notice that in the last interval, the only possibility is that Alice is in the bedroom and Bob in the kitchen, which is more precise than what can be inferred using only the last observations. The implemented algorithm thus successfully worked on the defined test case.

6 Conclusion and Perspectives

At the application level, our work has shown that it is possible to infer accurate location information with a minimum of sparse low level measurements. For instance, as proved by our illustrative example, our approach makes it possible to find out which rooms several known occupants of the home can be located in, even if only few of them can be identified through their mobile phone or RFID card and only very low level sensors and detector are used, only some rooms of the house are instrumented. The formalism and logical framework that we have defined multiple levels of genericity. In the Internet of Thing (IoT) domain, we can apply a similar approach to identify the status of an equipment (device, system, machine) through sparse observations of the equipment and of its environment.

On a more general level, we believe that our approach, including the modeling technique and algorithms can be applied to range of application domains. Characteristics of the target domains include the fact that information in these domains are organized as interrelated chunks of data and that it is known how modifying one chunk can affect chunks that are related to the chunk being modified.

On future work, this approach could also be extended to include probabilistic reasoning. This would allow to tackle the problem of imperfect sensors that can occasionally provide erroneous information, or to take into account the fact events and situations may occur with different probabilities.

References

1. Alchourron, C.E., Gardenfors, P., Makinson, D.: On the logic of theory change: partial meet contraction and revision functions. *J. Symb. Logic* **50**(2), 510–530 (1985)
2. Allen, J.F.: Maintaining knowledge about temporal intervals. *Commun. ACM* **26**(11), 832–843 (1983). <https://doi.org/10.1145/182.358434>
3. Allen, J.F.: Towards a general theory of action and time. *Artif. Intell.* **23**(2), 123–154 (1984). [https://doi.org/10.1016/0004-3702\(84\)90008-0](https://doi.org/10.1016/0004-3702(84)90008-0)
4. Doyle, J.: A truth maintenance system. *Artif. Intell.* **12**(3), 231–272 (1979). [https://doi.org/10.1016/0004-3702\(79\)90008-0](https://doi.org/10.1016/0004-3702(79)90008-0)
5. Fermé, E., Hansson, S.O.: *The AGM Model*, pp. 9–24. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-60535-7_3
6. Hoare, C.A.R.: An axiomatic basis for computer programming. *Commun. ACM* **12**(10), 576–580 (1969). <https://doi.org/10.1145/363235.363259>

7. Meyer, J.J.C.: Dynamic logic for reasoning about actions and agents. In: Minker, J. (ed.) *Logic-Based Artificial Intelligence*. The Springer International Series in Engineering and Computer Science, vol. 597, pp. 281–311. Springer, Boston, MA (2000). https://doi.org/10.1007/978-1-4615-1567-8_13
8. Siskind, J.: Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *J. Artif. Intell. Res. (JAIR)* **15**, 31–90 (2001). <https://doi.org/10.1613/jair.790>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.