Intel Serv Robotics (2013) 6:5–18
DOI 10.1007/s11370-012-0122-2

SPECIAL ISSUE

Temporal logic for process specification and recognition

Arne Kreutzmann · Immo Colonius · Diedrich Wolter · Frank Dylla · Lutz Frommberger · Christian Freksa

Received: 30 March 2012 / Accepted: 8 October 2012 / Published online: 2 November 2012

Abstract Acting intelligently in dynamic environments involves anticipating surrounding processes, for example to foresee a dangerous situation by recognizing a process and inferring respective safety zones. Process recognition is thus key to mastering dynamic environments including surveillance tasks. In this paper, we are concerned with a logic-based approach to process specification, recognition, and interpretation. We demonstrate that linear temporal logic (LTL) provides the formal grounds on which processes can be specified. Recognition can then be approached as a model checking problem. The key feature of this logic-based approach is its seamless integration with logic inference which can sensibly supplement the incomplete observations of the robot. Furthermore, logic allows us to query for process occurrences in a flexible manner and it does not rely on training data. We present a case study with a robotic observer in a warehouse logistics scenario. Our experimental evaluation demonstrates that LTL provides an adequate basis for process recognition.

Keywords Knowledge representation · Linear temporal logic (LTL) · Process recognition

This paper is a significantly extended and improved version of [18] presented at ECMR 2011. We have improved the interpretation of robot observations and we present a new experimental evaluation, based on an enhanced model checker implementation..

A. Kreutzmann · I. Colonius (✉) · D. Wolter · F. Dylla · L. Frommbe rger · C. Freksa
SFB/TR 8 Spatial Cognition, University of Bremen, Cartesium, Enrique-Schmidt-Str. 5, 28359 Bremen, Germany
e-mail: colonius@informatik.uni-bremen.de

D. Wolter
e-mail: dwolter@informatik.uni-bremen.de

1 Introduction

Mastering dynamic environments is a demanding challenge in autonomous robotics, involving recognition and understanding processes in the environment. Recent advances in simultaneous localization and mapping in dynamic environments build the basis for sophisticated navigation, but understanding processes goes even beyond. The ability to recognize and to understand processes allows a robot to interact with its environment in a goal-oriented fashion. For example, in processes that involve dangerous situations like the violation of safety zones, process understanding enables a robot to avoid dangerous situations in an anticipatory manner. But first of all, processes need to be represented in a way that fosters process understanding. Moreover, the representation should be seamlessly integrated with other high-level robot control tasks to ease the control flow.

We approach process understanding with linear temporal logic (LTL) ([29], see Sect. 3) which allows us to represent processes as logic formulas in a declarative manner. LTL is a slender knowledge representation language that recently has received increasing attention from the autonomous robotics community. The use of LTL in robotics has been advocated much earlier though [1]. For example, LTL has been used to specify controllers in a correct-by-construction manner [17]. LTL is widely used for motion planning from high-level specifications (e.g. [15,32,19]). Kloetzer and Belta [16] demonstrate the applicability to real robotic systems. Our motivation of using LTL is twofold. Firstly, we want to demonstrate that LTL specifications also provide an adequate basis for process recognition and understanding, supplementing existing approaches to robot control. Secondly, LTL allows a domain expert to describe processes of interest in a way that does not require knowledgeability of robot technology. LTL further provides an excellent basis

for flexibly querying the observations of the robot. It is then the task of the robotic system to turn a query into an effective observation and reasoning strategy.

In this paper, we focus on spatio-temporal processes, i.e., processes that are characterized by temporal patterns of movements in space. Spatio-temporal aspects are at the core of any process description and so this study achieves a high degree of generality. As scenario for our experimental evaluation we have selected warehouse logistics which is an interesting and relevant domain for studying spatio-temporal processes. In a warehouse, there is a steady flow of goods which are moved through space, establishing functional zones that are connected with certain types of storage processes (for example, admission of goods into a warehouse makes use of buffer zones to temporarily store goods). Note that these functional zones are not necessarily known a priori. Hildebrandt et al. [14] argue for use of autonomous robots as a minimally invasive means to recognize in-warehouse processes which, in turn provides the knowledge for optimizing the warehouse. The task of the robot is to recognize the storage processes that occur. However, a robot is generally not able to gather all potentially relevant information about a process and therefore needs to infer missing pieces of information, in particular identifying functional zones and their whereabouts.

The first contribution of this paper is to show that LTL offers adequate means for declaratively specifying processes in a way that fosters process recognition from robot observations. We demonstrate how a mobile observer can recognize various processes in a warehouse based on sensor perception backed up by a formal process specification. The second contribution of this work is to show that logic reasoning can be performed with the declarative process specifications and observations, enabling the robot sensibly to supplement missing pieces of information.

This paper is organized as follows: we first point out connections to existing work and we discuss reasons for choosing a logic-based formalism (Sect. 2). In Sect. 3, we briefly introduce LTL and summarize its important features. Thereafter, we describe our formalization of in-warehouse processes (Sect. 4) which consists of a domain axiomatization and an appropriate grounding of logic primitives. Section 5 presents our system realization, followed by an experimental evaluation (Sect. 6). We discuss our results (Sect. 7) and conclude with some final remarks (Sect. 8).

2 Related work

Many approaches have been used to tackle process recognition, which can roughly be categorized into learning approaches, probabilistic process descriptions, and logic-based declarative approaches.

Machine learning approaches such as Markov networks [5,20], Bayesian networks [36], supervised learning [3], or inductive logic programming [9] require a training phase before deployment. By contrast, we are particularly interested in mastering contexts in which no training data are available beforehand. Our aim is to enable querying the robot's observations using a flexible formal language for specifying process descriptions. Thus, any process to be recognized could be specified on the fly and does not need to be known beforehand; also queries to the system can be changed flexibly without need of relearning.

Declarative, logic-based formalisms enable us to pose queries flexibly. Utilizing logics in robotics dates back to the first appearances of AI robotic research (recall, for example, seminal work related to Shakey [28]). More recently, Mastrogiovanni et al. [25] have been using a logic-based approach integrating ontologies to recognize contexts in a ubiquitous robotics setting, which relates to our process recognition task. Mastrogiovanni et al. [24] introduced a new formal language to specify these contexts. In their framework, time is represented by a series of discrete time steps such that a formula holds at a given time instant. Computing time then increases exponentially with the number of time steps considered, such that only a limited number of time steps can be maintained. In the approach we present in this paper, we avoid this shortcoming by representing time explicitly on the level of the chosen logic formalism, namely linear temporal logics (LTL). This reduces the complexity and yields linear complexity with respect to the number of time steps as we will show in Sect. 3.2.

Moreover, formalisms based on LTL or its extensions neatly integrate into other LTL-based approaches to robot control such as motion planning or construction of controllers. Kress-Gazit et al. [17] propose a method for constructing controllers from an LTL formula and they determine that mastering state explosion is a key challenge, i.e., to develop techniques that avoid generating more states than feasible. This problem arises as LTL formulas are naturally evaluated over infinite time sequences, hence they potentially involve infinitely many states. One practical approach is to employ a receding horizon ([35,34,17], e.g.) which aims to cut off irrelevant future states. In our work we use a similar approach to interpret queries over the finite sequence of observations available to the robot. Ding et al. [8] propose a method to transform LTL specifications of processes into a control policy for Markov decision process, taking into account probabilities of successful action execution. This is accomplished in a way similar to model checking with a probabilistic temporal logic. Putting this into a more general context, probabilistic extensions of LTL, such as probabilistic temporal logic [7], are interesting. However, process detection with probabilistic logics requires the probabilities of process occurrence to be specified beforehand. In settings

like ours where no training data are available to determine the required probabilities, probabilistic logics are hence not suitable.

Model checking is widely used in software verification, but it has important applications in robotics, as well. For example, planning can be posed as a model checking task ([6,10,16], e.g.). Consider ϕ to be the specification of a plan to be fulfilled and M to be the set of all possible states a robot can be in. In this setting, verifying that M models ϕ means that we find a time linear sequence of robot states that meets the requirements of the specified plan. As we will show later, the same principle can be applied to process detection: the set of worlds M is then derived from sensor observations of the robot.

3 Linear temporal logic (LTL) for process detection

In classical propositional logic, formulas are evaluated with respect to a single fixed interpretation called world. Thus, a formula is either true or false. In order to acknowledge dynamic environments in which a proposition may hold for some limited time only, temporal logics utilize a set of worlds over which formulas are evaluated. Formulas may be satisfied in some worlds, but not in others. Linear temporal logic is a modal logic that extends propositional logic by a sequential notion of time. A formula ϕ in LTL is defined over a finite set of propositions with the usual logic operators ($\wedge$, $\vee$,$\neg$, $\rightarrow$). The temporal component is established by an accessibility relation R that connects the individual worlds (also called states with LTL) over which formulas are interpreted. In linear temporal logic, the relation R is a discrete linear ordering. We say that a world W is a future world of V if $(V, W) \in R$, i.e., W is reachable from V by R. LTL defines three unary modal operators on the basis of R:

$\circ\phi$	(next)	ϕ holds in the following world
$\Box\phi$	(always)	ϕ holds in the current world and in all future worlds
$\Diamond\phi$	(eventually)	ϕ holds in some future world ($\Diamond\phi \leftrightarrow \neg\Box\neg\phi$)

One important reasoning task in logic is model checking: given a specification ϕ and a model which valuates the logic primitives, does the model satisfy ϕ?

3.1 Process recognition as model checking

We describe a process by an LTL formula ϕ. Then, a process is said to be recognized if a model derived from the observations of the robot satisfies ϕ. Thus, model checking matches logic predicates with observations. Usual techniques for LTL model checking are based on translation of the formulas to either ω-automata or Büchi-automata. These automata are then used to process an infinite model. However, in process detection we are involved with finite models. For any given time point one can decide whether a complete process has been observed or not.

3.2 Computational complexity of model checking

Sistla and Clarke [31] show that the model checking problem of LTL is PSPACE-complete, but various fragments have a significantly lower complexity. Bauland et al. [4] investigated various fragments of LTL and found tractable subsets with time complexity as low as NLOGSPACE-complete. Efficient subclasses either exclude the eventually operator or they do not allow the Boolean and. However, both operators add important expressiveness to process descriptions. Our formalization of warehouse processes detailed in Sect. 4.6 involves both operators. The resulting subclass of model checking which involves only $\Diamond$, $\wedge$, and $\neg$ is NP-complete [4,31].

In a survey about complexity of temporal logic model checking, Schnoebelen [30] describes the influence of various factors. If the length of formulas is fixed, complexity of model checking is in NLOGSPACE with respect to model size $|M|$. By contrast, if the model is fixed, then the complexity is in PSPACE with respect to varying length of the formulas. Lichtenstein and Pnueli [21] show that model checking can be done in $2^{O(|\phi|)}O(|M|)$, i.e., model checking is linear-time with respect to the size of the model. This is an important result since in applications like process detection the model size grows with the amount of observations, but the length of the query formulas is fixed, assuming fixed process descriptions. In other words, the exponential growth with respect to formula length does not apply.

We consider an important variety of model checking. Consider formulas which disjunctively combine sub-formulas which only vary in one atom, i.e., formulas which can be written $\bigvee_{a\in A}\phi(a)$, whereby A denotes a set of atoms. If all $\phi(a)$ are within an NP-complete fragment of LTL model checking, the complete formula is also in NP as one can non-deterministically select a clause from the disjunction in polynomial time and then continue with model checking. Morgenstern and Schneider [27] pursue a similar idea by introducing oracle variables which "[...] may represent 'angelic' nondeterminism that may be resolved in favor to satisfy the specification".

We can thus apply LTL model checking to formulas that involve an extensional quantifier ranging over a set of atoms without increasing computational complexity further. This observation is important in our context since we are interested in querying for process occurrences and queries naturally involve unknowns. For example, one would rather query whether a good was moved to some place within the storage area, rather than querying whether a specific

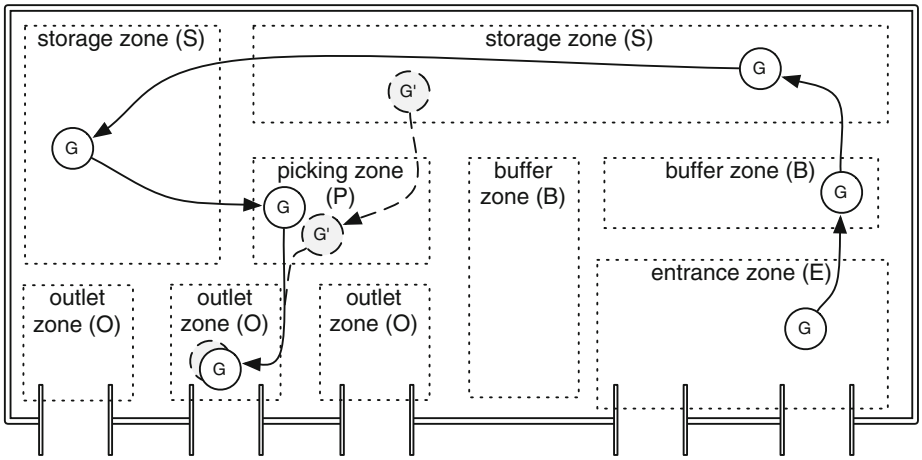


Fig. 1 A warehouse, its functional zones, and typical movements of different goods (G, G')

good G was moved to a specific location L. With respect to computational complexity of querying we note that the procedure can be carried out in NP given that the set of atoms to consider is fixed, i.e., our domain is not expanding. Considering our warehouse domain we naturally have a fixed set of locations but a potentially growing set of goods. However, within a limited amount of time the amount of goods visiting a warehouse can be regarded to remain constant.

4 Specification and interpretation of in-warehouse processes

In the following, we describe a case study of warehouse logistics in which a mobile robot observes processes in a warehouse. The robot can later be queried by a logistic expert who is involved with improving storage strategies. We use LTL to describe relevant storage processes and their functional components. Both temporal and spatial primitives used in the logic are grounded in the observations of the robot. We conclude this section by a small example.

4.1 Scenario

We address the problem of understanding so-called chaotic or random-storage warehouses, characterized by a lack of predefined spatial structures, that is, there is no fixed assignment of storage locations to specific goods. Thus, storage processes are solely under the responsibility of the warehouse operators and basically are not predictable: goods of the same type may be distributed over various locations and no database keeps track of these locations. This makes it a hard problem for people aiming to improve the internal storage processes. We are interested in representing the spatio-temporal change that occurs in the warehouse, but we are not interested in tracking individual movements. Therefore, we can assume the environment to be piecewise static.

On a conceptual level, storage processes are defined by a unique pattern [33]: on their way into and out of the warehouse, goods are (temporarily) placed into functional zones which serve specific purposes (see Fig. 1). All goods arrive in an entrance zone (E). From there, they are picked up and temporarily moved to a buffer zone (B) before they are finally stored in the storage zone (S). This process is called 'admission'. Within the storage zone 'redistribution' of goods can occur arbitrarily. When 'taking out' goods, they are first moved from the storage zone to the picking zone (P) from where they are taken to an outlet zone (O), before being moved out of the warehouse.

A mobile robot observing such a warehouse is not able to perceive these zones directly, as they are not marked. For all zones we know that they exist (that is, that such regions are used within the storage operations), but neither their concrete spatial extents nor the number of their occurrences is known. This information solely depends on the dynamic in-warehouse storage processes. The robot can detect and identify goods and it can estimate their position. We thus face the challenge that for detecting concrete storage processes we need to rely on knowledge about functional zones which is not yet available. For example, if a robot perceives a good at three different locations the process interpretation largely depends on the zones of the locations. If all locations are in the storage zone, a redistribution may have occurred, whereas if all locations are in different zones an admission or a take-out process may have occurred.

4.2 Formalizing the warehouse scenario

In this section we explain the formalization of processes and general background knowledge in terms of spatio-temporal integrity constraints like, for instance, the fact that objects can only be at one location at a time. To this end we need to compose LTL formulas which capture the characteristics of spatio-temporal processes. These formulas serve as axioms

and are used to enforce a sensible interpretation of the observations of the robot. To begin with, observations are mapped in a spatio-temporal grounding process to primitives of our logic. Our formalization is based on the following set of primitives:

Goods: a set $\mathcal{G} = \{G_1, \ldots, G_n\}$ of goods constitutes the entities that move in space over time and determine the dynamics of the scenario. They are observable by the robot and their position can be estimated.
Locations: a location is a property of a good which remains the same when a good is not moved. During spatio-temporal grounding, position estimates are abstracted to a discrete set of locations. For a spatially restricted scenario the set of locations $\mathcal{L} = \{L_1, \ldots, L_m\}$ is finite.
Zones: the warehouse scenario involves functional zones $\mathcal{Z} = \{E, B, S, P, O\}$ as described in Sect. 4.1. The extent of a zone is defined by the set of locations it contains. Zones are considered to be fixed in our scenario, but their extent is a priori unknown to the reasoning system.

4.3 Atomic propositions for spatio-temporal processes

Modeling with LTL involves devising a finite set of atomic propositions which capture relevant facts about the state of the world. Atomic propositions can either be determined by interpreting observations of the robot or by logic inference. We utilize the following atomic propositions which we denote in a predicate style for ease of readability, i.e., the atom $\text{at}(G, L)$ stands for $|G| \cdot |L|$ atoms, one per combination of good G and location L.

- $\text{at}(G, L) \Leftrightarrow$ good G is at a location L.
  This type of atom is data-driven, that is, its value can directly be obtained from sensor observations of the robot. Proposition $\text{at}(G, L)$ holds if and only if a good G is known to be at location L. Truth of this proposition can thus change over time if a good is moved.
- $\text{in}(L, Z) \Leftrightarrow$ location L is contained in a zone Z.
  As the set of locations is generated at runtime, $\text{in}(L, Z)$ also depends on sensor perceptions. The interpretation of $\text{in}(L, Z)$ remains constant over time.
- $\text{close}(L_1, L_2) \Leftrightarrow$ two locations L_1, L_2 are close to one another.
  We use closeness as a central concept to distinguish different zones. $\text{close}(L_1, L_2)$ remains constant over time.

4.4 Spatio-temporal grounding

In LTL, time is represented as a sequence of independent worlds. A temporal interpretation can thus be achieved by sampling the observations of the robot. To this end, we make use of the perception loop of the robot. During each cycle, sensors are read and localization and mapping are updated. The updated information is then used to determine which of the atomic propositions currently holds. Since our domain does not require us to state that nothing has changed, we can reduce the set of worlds emitted by temporal grounding. A new world is only generated if the interpretation of at least one atomic proposition changes.

One central task of spatio-temporal grounding is the robust interpretation of position estimates $(x, y) \in \mathbb{R}^2$ to discrete locations $L_i \in \mathcal{L}$. Naturally, position estimates are subject to noise and may vary over time even if an object does not move. Additionally, by keeping the size of the set of locations minimal, we can minimize the set of atomic propositions and thereby limit the size of our formulas. This can be accomplished by updating the set of locations at runtime, adding new locations only if necessary. In our implementation we apply a clustering approach that takes uncertainty of estimates into account (see Sect. 5.2). It provides us with a compact and robust interpretation, but other methods would be possible too. A requirement is, however, that the mapping from positions to locations is stable over time, i.e., if a good G is said to be located at L_i then this interpretation shall not be revised when observations are integrated into the localization procedure. With $\mathcal{L}$ available, $\text{at}(G, L)$ can be directly derived for every time step. Furthermore, $\text{close}(L_1, L_2)$ is valuated by applying a metric and checking whether the distance between L_1 and L_2 is below a certain threshold (in the experiments in this paper, we use an Euclidean distance of 1 m). The propositions $\text{in}(L, Z)$ can be set if knowledge about zones is available a priori, otherwise they need to be inferred by reasoning.

4.5 Spatio-temporal integrity constraints

Commonsense knowledge about spatio-temporal processes in our domain is captured by the following set of axioms which enforce a sensible interpretation of data available. While processes and related queries can be freely specified, axioms remain the same over any process detection task. Explicating this knowledge in axioms separately allows us to keep the process specification simple and intuitive. We define the following four axioms:

- Locations are fixed, i.e., if two locations are close to one another they are always close to one another.

$$A1_{L_i, L_j} = \text{close}(L_i, L_j) \rightarrow \Box\text{close}(L_i, L_j) \quad \text{(A1)}$$

- A good G can only be at one location at a time.

$$A2_G = \Box \bigwedge_{L_i \neq L_j} \neg\big(\text{at}(G, L_i) \wedge \text{at}(G, L_j)\big) \quad \text{(A2)}$$

Axioms (A1) and (A2) describe common-sense spatio-temporal constraints. Their fulfillment is guaranteed by the spatio-temporal grounding process and therefore can be omitted in the reasoning process. The next two axioms need to be addressed explicitly during model checking:

– A location $L \in \mathcal{L}$ always belongs to the same zone and is exactly in one zone $Z \in \mathcal{Z}$. In other words, zones are static and do not overlap.

$$A3_L = \bigvee_{Z \in \mathcal{Z}} \Box\Big(\text{in}(L, Z) \wedge \big(\bigwedge_{Z' \in \mathcal{Z} \setminus \{Z\}} \neg\text{in}(L, Z')\big)\Big) \tag{A3}$$

– Locations in different zones are not close to one another, that is, zones are at least some minimum distance apart. We note that it is still possible that two locations which are not close to one another can belong to the same zone (multiple occurrences of zones).

$$A4_{L_i,L_j} = \Box\big(\text{close}(L_i, L_j) \rightarrow \bigvee_{Z \in \mathcal{Z}} (\text{in}(L_i, Z) \wedge \text{in}(L_j, Z))\big) \tag{A4}$$

In the following we use $\mathcal{A}'$ to refer to axioms (A3)–(A4) that are essential for process recognition. In conduction with all propositions 'close' and 'in' which are static over all worlds [also constituted by (A1) and (A3)] this forms the set

$$\mathcal{B} = \mathcal{A}' \cup \bigcup_{L_i,L_j \in \mathcal{L}} \text{close}(L_i, L_j) \cup \bigcup_{L \in \mathcal{L}, Z \in \mathcal{Z}} \text{in}(L, Z) \tag{1}$$

that we call the background knowledge of the warehouse domain.

In situations where further knowledge about zones is available, the axioms can be modified to accommodate such a priori knowledge, for example by adding appropriate propositions $\text{in}(L, Z)$ to the set of axioms. In our evaluation we make use of such modifications to $\mathcal{B}$ in order to study the effectiveness of inferring zone membership by reasoning.

4.6 In-warehouse processes

We now formalize in-warehouse processes. In particular, we define admission, take-out, and redistribution of goods. All processes are specified using the following schema:

$$\text{start condition} \wedge \Diamond(\text{next state condition} \wedge \Diamond(\ldots)). \tag{2}$$

The schema solely captures the characteristic states of a process. This ensures a robust detection of processes which can vary in the level of detail.

– Admission: a good G is delivered to the warehouse's entrance zone E and moved to the storage zone S via the buffer zone B. For all $G \in \mathcal{G}$ and $L_i, L_j, L_k \in \mathcal{L}$:

$$\begin{aligned}\text{Admission}_{G,L_i,L_j,L_k} = {} & \text{at}(G, L_i) \wedge \text{in}(L_i, E) \\ & \wedge\Diamond\Big(\text{at}(G, L_j) \wedge \text{in}(L_j, B) \\ & \wedge\Diamond\big(\text{at}(G, L_k) \wedge \text{in}(L_k, S)\big)\Big)\end{aligned} \tag{3}$$

– Take-out: a good G is moved from the storage zone S to the outlet zone O via a picking zone P. For all $G \in \mathcal{G}$ and $L_i, L_j, L_k \in \mathcal{L}$:

$$\begin{aligned}\text{Takeout}_{G,L_i,L_j,L_k} = {} & \text{at}(G, L_i) \wedge \text{in}(L_i, S) \\ & \wedge\Diamond\Big(\text{at}(G, L_j) \wedge \text{in}(L_j, P) \\ & \wedge\Diamond\big(\text{at}(G, L_k) \wedge \text{in}(L_k, O)\big)\Big)\end{aligned} \tag{4}$$

– Redistribution: a good G is moved within the storage zone S. For all $G \in \mathcal{G}$ and $L_i, L_j \in \mathcal{L}, i \neq j$:

$$\begin{aligned}\text{Redistribution}_{G,L_i,L_j} = {} & \text{at}(G, L_i) \wedge \text{in}(L_i, S) \\ & \wedge\Diamond\big(\text{at}(G, L_j) \wedge \text{in}(L_j, S)\big)\end{aligned} \tag{5}$$

4.7 Inferring functional zones

The axioms and the process specifications make use of functional zones like entrance or buffer. While some zones may be known beforehand, others are not known and need to be inferred. Zones are characterized by the functional role they take, for example, an outlet zone is the set of locations in which a good can be seen last before it is taken out of the warehouse. Thus, identifying zones is the task of identifying a set of locations close to one another which all take the same functional role in the storage processes observed. This task can be viewed as model checking: do the observations provide a model for a hypothesis that a set of locations takes a specific functional role? During model checking variables in a process description are instantiated with observations. This includes that the locations involved in process descriptions are assigned to zones. In other words, inference of functional zone happens naturally during model checking. For example, if trying to verify that an admission has taken place, at least one location L_e is required to belong to an entrance zone. Axiom (A4) further enforces that all locations close to each other belong to the same zone. Technically speaking, zone membership is ruled by the transitive closure of the close

relation on locations. This leads to an interpretation that is consistent with all processes detected.

4.8 Histories and complex process queries

One piece of good can participate in many processes. Goods enter a warehouse in an admission process, they possibly get redistributed a couple of times before they eventually leave the warehouse in a takeout process. We call the sequence of processes a good participates in the history of the good. In our domain histories naturally begin with an admission and end with a takeout. Analogous to process detection, histories can also be detected in an atomic manner, i.e., LTL allows us to pose complete histories as a single query. This can be accomplished by using the same basic schema as shown in Eq. (2). We give three examples of complex queries to highlight the generality of the LTL-based approach to process detection by model checking:

– Has a good G been moved back and forth?

$$Q_1 = \mathrm{at}(G, L_i) \wedge \Diamond\big(\mathrm{at}(G, L_j) \wedge \Diamond \mathrm{at}(G, L_i)\big) \wedge \overbrace{\neg(\mathrm{at}(G, L_i) \wedge \mathrm{at}(G, L_j))}^{L_i \neq L_j} \tag{6}$$

– Has a good G been redistributed k or more times?

$$Q_2 = \phi_{L_{i,1}} \wedge \underbrace{\Diamond(\phi_{L_{i,2}} \wedge \Diamond(\phi_{L_{i,3}} \wedge \Diamond(\cdots)))}_{k \text{ nestings}} \tag{7}$$

with

$$\text{Redistribution}_{G,L_k,L_l} = \overbrace{\mathrm{at}(G, L_k) \wedge \mathrm{in}(L_k, S)}^{=\phi_{L_{i,j}}} \wedge \Diamond \overbrace{(\mathrm{at}(G, L_l) \wedge \mathrm{in}(L_l, S))}^{=\phi_{L_{i,j+1}}}$$

– Do two goods G_i, G_j with $G_i \neq G_j$ that have been observed together once remain co-located?

$$Q_3 = \big(at(G_i, L_k) \wedge at(G_j, L_l) \wedge \mathrm{close}(L_k, L_l)\big) \rightarrow \Box\Big(\overbrace{\bigvee_{\substack{L_m, L_n \\ \mathrm{close}(L_m, L_n)}} \big(\mathrm{at}(G_i, L_m) \wedge \mathrm{at}(G_j, L_n)\big)}^{G_i, G_j \text{ are observed together...}} \vee \underbrace{\neg \bigvee_{L_o} \big(\mathrm{at}(G_i, L_o) \vee \mathrm{at}(G_j, L_o)\big)}_{\text{...or neither is observed}}\Big) \tag{8}$$

By conjunctively joining process specifications, arbitrarily complex queries can be stated. Prefixing a process in a conjunctive query by the eventual operator ($\Diamond$) states, that the process may happen any time, independent of the other processes. Joint queries are important to enable reasoning across histories. By conjunctively combining several history queries and prefixing them by $\Diamond$ we obtain a single query for the existence of a model that satisfies all processes involved (joined histories). In particular, this leads to a jointly compatible interpretation of functional zones. For example, during individual queries in one of the solutions a location might be interpreted to be a buffer area while during querying for a different good it is interpreted as part of an entrance. In the case of jointly querying for both histories, the same location cannot be part of different zones as of axiom (A3). Therefore, we only obtain histories as a result that satisfy a process specification using the same interpretation of location-zone membership. This results in more robust interpretation but comes at the cost of higher computing time due to increased formula size.

4.9 Example

A good G enters the warehouse and is stored in the entrance zone E at position L_1 at time t_0. Between t_1 and t_2 the good is moved to a location L_2 and between t_2 and t_3 the good is moved further to L_3. Let us assume that this process is observed as follows: we perceive G to be at L_1 at t_1, at L_2 at t_2 and at L_3 at t_4. Furthermore, all these locations are not close to one another. See Fig. 2 for a depiction and the logic interpretations—to ease understanding the worlds are labeled by time points. These observations constitute a model that satisfies (3), i.e., the observed process is an admission that starts in world t_1 and ends in world t_4. By inference it follows that location L_2 is contained in the buffer zone and L_3 is contained in a storage zone. Note that detected start and end times differ from the real admission times: while the admission takes place from t_0 to t_3, we detect it from observations t_1 to t_4; this is due to incomplete observation of the world.

5 System realization

Our implementation of logic-based process recognition essentially consists of three parts: perceptions by the robot, their symbolic interpretation and process understanding using the high-level process model. The system architecture is shown in Fig. 3. The first part, perception, is integrated with our robot control software and its main objective is to localize the robot and to provide an up-to-date map of the changing warehouse. Essentially, we utilize a feature-based SLAM to map the environment, using tag-based good

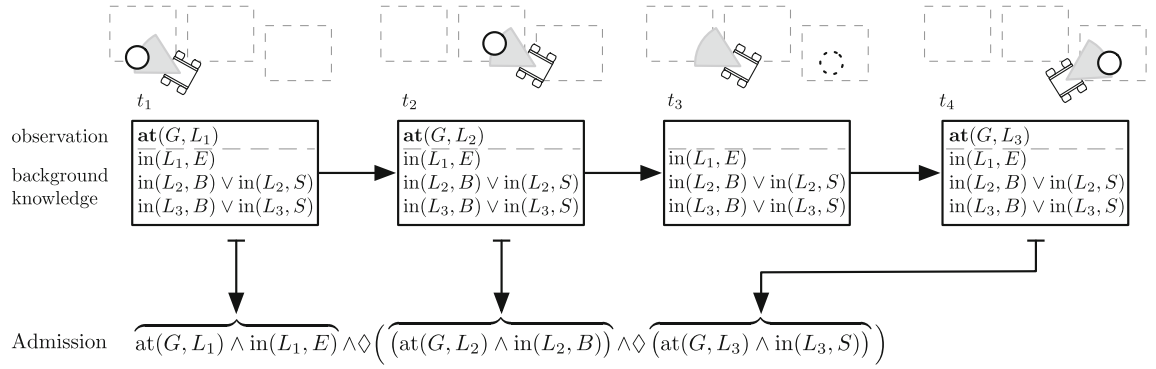


Fig. 2 Example: model checking for an admission process of good G (only the relevant assertions for each world $t_{1...4}$ are shown). $\text{in}(L_1, E)$ is background knowledge, also it is known that locations L_2 and L_3 are either part of the buffer zone (B) or the storage zone (S) but not close to one another so that they do not have to belong to the same zone. From this admission refined knowledge about the buffer and storage zones can be inferred: $\text{in}(L_2, B) \wedge \text{in}(L_3, S)$

identification. Perceptions are then lifted to the symbolic level. By evaluating the posterior probability of changes in landmark positions at each time step we determine whether a good was moved and can update the map accordingly, keeping track of all re-locations. During symbolic grounding the position estimates obtained from the robot map are also clustered to discrete qualitative locations that are then employed to describe the trajectory of good movements. We refer to the output of the symbolic grounding stage as qualitative observations. Process recognition is realized by the symbolic reasoning component that matches qualitative observations against the process descriptions or process queries, supplementing the qualitative observations by inferred knowledge.

5.1 Perception: localizing and mapping goods

Localization and mapping of goods is realized as a feature-based SLAM using visual tags attached to both goods and the environment (see Fig. 5). We use the ARToolKit software[1] for identifying tags in camera images. This toolkit provides us with a tag identifier and with a 3D projection matrix that estimates the tag position and its orientation relative to the camera. We project this transformation to the 2D ground plane in order to obtain a bearing-and-distance estimate that is used with the SLAM system. To this end, we determine a measurement model for our camera. This model is coarsened to mimic RFID-scanners that would be typical in an industrial context. By only working with tags positioned in the same height, a simple projection suffices to calculate the 3D to 2D transformation. ARToolKit also provides a quality of recognition which we use to gate tag recognition, discarding any identifications with less than 80 % quality.

[1] http://www.artoolkit.sourceforge.net/

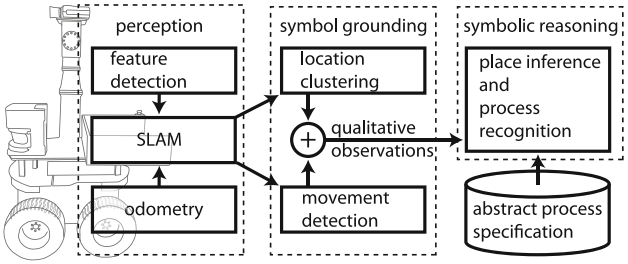
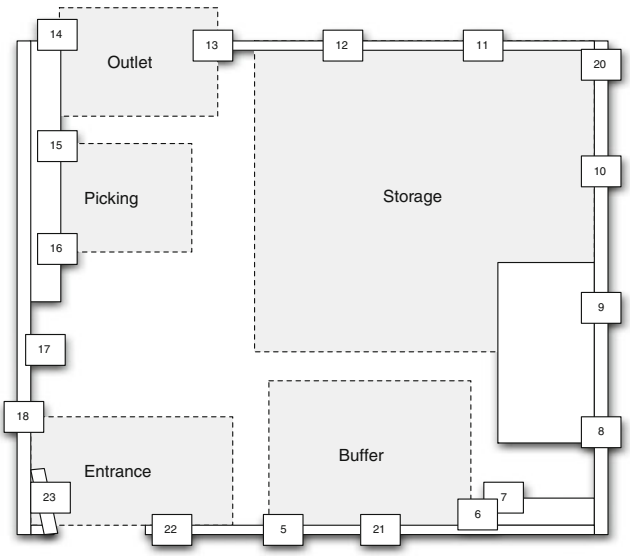


Fig. 3 System architecture of robotic platform and reasoning

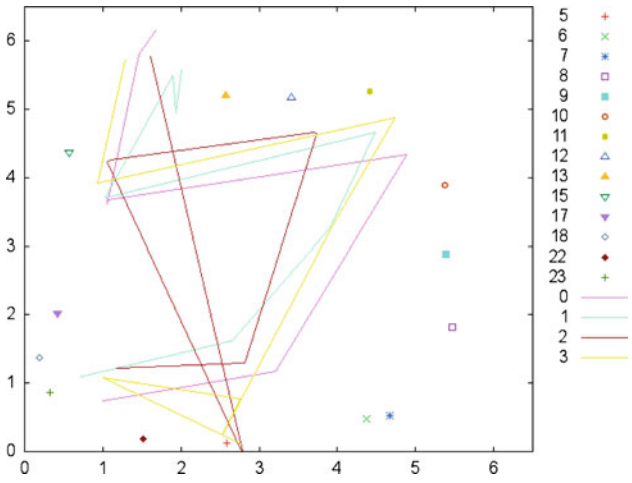
Feature-based SLAM is accomplished by a modified version of the TreeMap system [11] for simultaneous localization and mapping in static environments. TreeMap estimates the position of 2D landmarks by a least square approach, assuming a Gaussian noise model for odometry and observations. Originally, TreeMap does not grant access to covariances. We extended TreeMap to provide us with covariances for position estimates of landmarks. This allows movement detection in an uncertainty-sensitive manner by determining the Mahalanobis distance between the position of an observed landmark and its position given by the map, using both covariance of observations according to the measurement model and covariance of the position estimate. A movement is said to be detected if the Mahalanobis distance exceeds a fixed threshold of 1.9. Moved goods are re-entered into the SLAM system using a new identifier. By keeping track of the different identifiers used to refer to a single landmark we can reconstruct its trajectory.

5.2 Symbol grounding: from perception to qualitative observations

Position estimates in the robot map are clustered immediately before the logic-based process detection is invoked in order



(a) Warehouse layout with positions of static landmarks.



(b) SLAM map for experimental run I consisting of 4 trajectories for goods $(0, 1, 2, 3)$ and static landmarks $(5..23)$

Fig. 4 Experimental setup: warehouse in the lab (6.12×7 m). **a** Warehouse layout with positions of static landmarks. **b** SLAM map for experimental run I consisting of four trajectories for goods (0,1,2,3) and static landmarks (5..23)

to obtain a small and robust set of locations which describe positions of goods in LTL. We use a straightforward centroid clustering that iteratively processes position estimates generated by the SLAM system. A new cluster is generated whenever a position does not fit into any cluster already established. For every cluster we generate a location L and valuate the atoms $\text{at}(G, L)$ and $\text{close}(L, L')$ (distance between locations less than 1 m) accordingly. Clusters are limited in size to a circle of 0.25 m radius as they represent single qualitative locations only. The iterative clustering method may not yield the most sensible interpretation of locations, but it ensures that the assignment from positions to clusters and thus locations will not be revised if new observations are available. This avoids detection of spurious movements and ensures monotony of the reasoning process (cp. Sect. 4.4). To study the effects of the autonomous clustering method, we employ an additional method that uses pre-defined centroids and allows us to test ground truth data for the centroids.

All in all, we obtain a time-discrete sequence of observations, e.g., $t_0 : \{\text{at}(a, l_1), \text{at}(b, l_2)\}$, $t_1 : \{\text{at}(b, l_1)\}$, and so on. To construct the sequence of qualitative observations, repetitive time points are collapsed into a single qualitative state, i.e., we omit all observations which share the same set of $\text{at}(\cdot)$ atoms as the preceding observation.

5.3 Symbolic reasoning: process understanding with qualitative observations

For performing process recognition we utilize the modeling language of answer set programming (ASP). Based

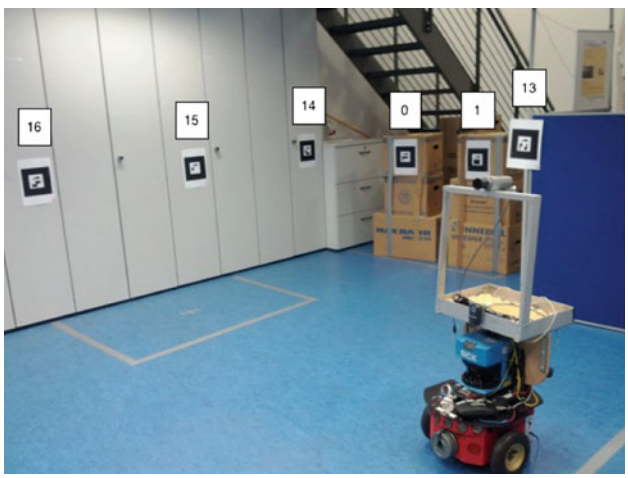


Fig. 5 Warehouse mockup equipped with AR-tags and a Pioneer 2-DX (see also Fig. 4a)

on its roots in logic-based knowledge representation and non-monotonic reasoning, databases, satisfiability testing and logic programming, ASP offers high-performance tools while providing us with a rich yet simple modeling language. The semantics of ASP is based on the stable model semantics [22,23].

Linear temporal logic semantics can easily be achieved with ASP. First, qualitative observation atoms are attributed with the world in which they hold, for example $\text{at}(G, L)$ becomes $\text{at}(W, G, L)$. Second, the modal operators are realized. The next operator is realized as a preposition on worlds, i.e., we add $\text{next}(W_i, W_{i+1})$ if W_{i+1} directly follows W_i.

Table 1 Scenarios evaluated, their characteristics with respect to problem size, and computation time for the symbolic process recognition

Scenario	#Goods (histories)	#Processes	Duration (m:s)	#Observations	Largest joined history[1]
A	1	2	5:24	5	0.0 s (±0.0)
B	4	8	8:08	70	1.0 s (±0.1)
C	4	8	10:42	125	8.5 s (±0.8)
D	4	8	11:56	188	41.1 s (±4.3)
E	4	8	13:08	192	55.2 s (±6.8)
F	4	8	14:35	71	0.8 s (±0.5)
G	4	8	15:48	95	3.5 s (±0.5)
H	4	10	10:35	197	71.8 s (±11.1)
I	4	10	18:03	107	3.5 s (±1.6)
J	4	10	18:38	177	37.1 s (±6.5)
K	11	24	29:00	326	23.7 s (±2.4)
L	12	32	34:06	473	152.7 s (±21.7)

[1] Averaged over varying zone knowledge: full, partial, and no previous knowledge
For a definition of joined history see Sect. 4.8

Future is realized as a recursively defined preposition utilizing the next operator. Then, process specifications and queries are rewritten accordingly. Axioms are modeled as constraints and free variables occurring in queries are represented by choice rules in ASP. We use GRINGO for grounding and CLASP as ASP solver [12,13].[2]

In general, one set of observations can be interpreted differently in terms of which histories could have occurred. Consider the example of moving a good from A to B and further to C. This clearly satisfies the model $\text{Redistribution}_{G,A,B} \wedge \text{Redistribution}_{G,B,C}$, but it also satisfies $\text{Redistribution}_{G,A,C}$. The latter interpretation ignores the observation that the good visited location B. In such cases we select the maximal model in the sense of selecting the history that involves the largest number of processes—this can also be performed by the ASP solver. The example is thus interpreted as two redistributions.

6 Experiments and evaluation

In our experimental setting we simulate warehouse processes in our lab. We measure how many histories, i.e., chains of processes per good, can be identified correctly. The numbers are further detailed to study the ability of the symbolic component to counteract absence of process knowledge. Also, we analyze the computing time of the symbolic reasoning component.

6.1 Experimental setup

Our experimental robot platform consists of an Active Media Pioneer 2-DX (differential drive) controlled by a top-mounted laptop and equipped with a SONY DFW SX900 (approximately 160° FOV) camera that delivers 7.5 frames per second.

We simulate a warehouse that consists of five dedicated zones (entrance, buffer, storage, picking, outlet) as depicted in Figs. 1 and 4a. Each good is labeled with a unique visual tag as shown in Fig. 5 (rectangular shapes on paper sheets). For tag identification, we rely on the ARToolKit. We distribute 17 tags as static landmarks over the environment in order to ease robot localization.

One run of the experiment consists of a series of movements of goods between the zones while our robot is monitoring the environment. The location of all tags is determined and we record which processes happen to obtain ground truth data for evaluation. For each of the 12 scenarios performed, the robot was manually driven around the test environment until each landmark has been seen at least once to ensure a robust localization. Then, we steered the robot in random courses, while we moved boxes through the lab, simulating the previously defined logistic processes (Sect. 4.6). The duration of a single run was between approximately 5 and 34 min in which we moved one to 12 goods through the warehouse, resulting in two to 32 detectable processes (admission, redistribution, takeout) per run. Details are shown in Table 1. Goods were moved between zones while not covered by sensor surveillance to comply with Axiom (A2) in Sect. 4.2. Data gathered by the robot are processed as described above to obtain good trajectories (see Fig. 4b for an example depicting the movement of our goods) that are then interpreted in terms of qualitative observations and passed to the symbolic process recognition to recognize histories of all goods. We say that a history is correct if all detected processes and their temporal order matches with the ground-truth.

[2] As provided at http://www.potassco.sourceforge.net

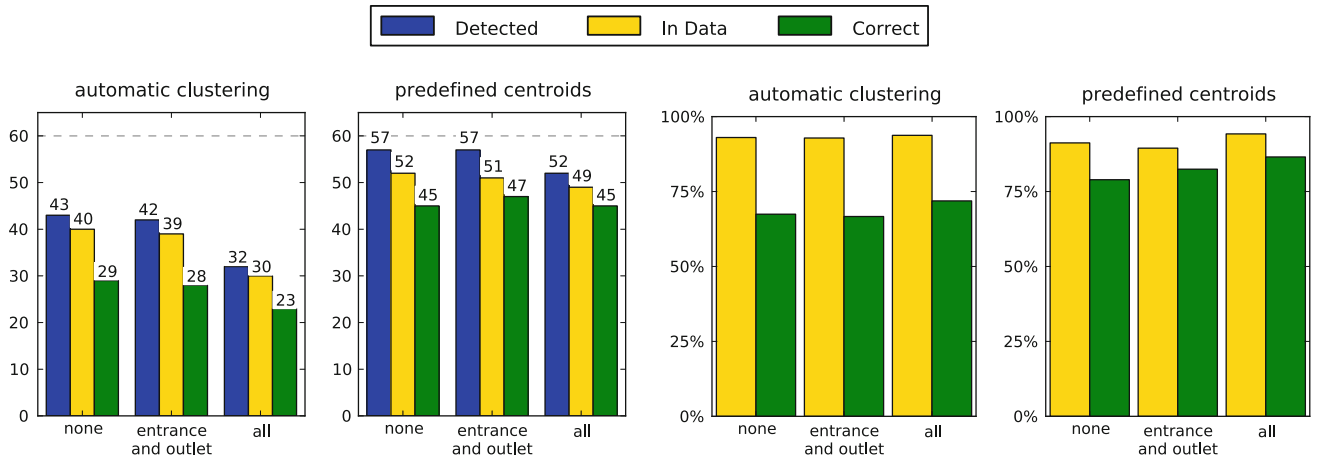


Fig. 6 Results from the experimental evaluation. **a**: showing the number of histories detected, how many are supported by the data, and how many are correctly recognized. **b** Percentage of histories supported by the data versus correctly detected histories while optimizing for maximal history length; the columns are normalized by detected histories. **a** Evaluation results, **b** relative quality of detected histories

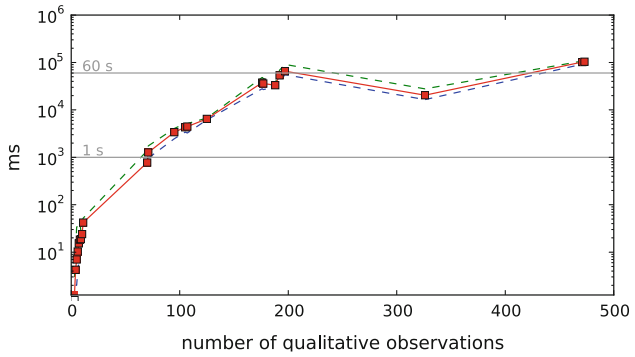


Fig. 7 Plot of the computing times (in log scale) for symbolic process recognition versus number of qualitative observations

6.2 Evaluation

For the evaluation we perform 12 experimental process scenarios, resulting in a total of 60 histories, one for each ware. We record all intermediate processing results. Additionally, we record ground truth information. The characteristics of these scenarios with respect to problem size are shown in Table 1. In the column '# observations' we give the number of qualitative observations obtained by the automatic clustering. The computing times for the symbolic processing also refer to the fully automatic clustering method. Figure 7 further presents the computing times for queries obtained with respect to the number of qualitative observations as this is the essential factor of the computing time.

We determine the total number of correctly identified histories across all scenarios, breaking up the numbers into the availability of zone knowledge (all location to zone mappings known, only entrance and outlet known, no zone known) and by the spatial grounding method used (automatic clustering vs. pre-defined centroids). Figure 6a shows the results obtained and graphically presents the percentage of histories recognized depending on these factors. Furthermore, the plots in Fig. 6 also show the relative amount of correct histories verifiable by qualitative observations in the data. To obtain this measure, we query the qualitative observations using the ground-truth.

7 Discussion

We first consider quality of process recognition. Looking at Fig. 6, the "in data" bars represent the relative amount of histories correctly verifiable matching the qualitative observations against ground truth. In other words, this bar represents how well the real-world process is captured by the robot observations, their symbolic interpretation, and the overall process description. This bar can thus be considered a gold standard for the actual process recognition algorithm. There are several reasons why the gold standard does not reach the 100 % mark, for example good histories could not be reconstructed correctly (small movements easily remain unrecognized by the mapping component), or the robot may have simply overlooked essential information. We obtain recognition rates of around 61 % in case of automatic clustering and 84 % in case of predefined centroids for this gold standard, which indicates that the symbolic process descriptions, the qualitative interpretation of sensor data, and the integration with the robotic system provides us with an adequate foundation. The correct bars in Fig. 6 present the absolute/relative correct recognitions achieved by our process recognition algorithms. Naturally, the performance is less than that of the gold standard and we observe a difference in performance comparing the automatic clustering

method against clustering with pre-defined centroids. This difference indicates the importance of a sensible spatial grounding and motivates further research to obtain more sophisticated automated methods. As expected, we observe that with increased background knowledge the relative number of histories matching the ground truth increases while the total number of detectable processes decreases. The reason for this is due to the fact that providing more background information restricts the way the data can be interpreted, leading to fewer interpretations that meet a process description.

In some scenarios when all regions are known it occurred that some locations are not within any zone, thereby violating Axiom (A3) and hence inhibiting recognition at all. This is especially true in the case of the automatic clustering as can be seen by the drop in the case when all regions are known. Overall the increase in background knowledge reduces the amount of false positives while having little impact on the number of correctly detected histories.

The most important observation is however that the relative number of correctly identified histories, i.e., how many from the detected histories are correct, is hardly affected by the amount of background knowledge available about zone membership. In case of pre-defined cluster centers the average relative recognition rate is 83 % whereas for the case of automatic clustering the average relative recognition is 69 %. Missing zone membership information is compensated for by logic reasoning during model checking which determines the unknown variables sensibly. In other words, the inference process is capable of supplementing all missing zone membership information to the process recognition process. This demonstrates that a logic-based approach is a valuable contribution to process recognition methods.

We note that these results confirm a previous study with respect to the overall conclusion, absolute recognition rates have improved though (cp. [18]), in the case of pre-defined clusters from 68 % achieved previously to about 76 % in this study (for automatic clustering from 42 to 44 %). The relative recognition rates, i.e., how many of the detected histories are correct ones, have improved even more: in case of pre-defined clusters from 73 to 83 % and in case of automatic clustering from 57 to 69 %. This improvement is due to three changes: first, we changed the robot hardware from a Pioneer 3-AT (four wheel skid steering drive) to a Pioneer 2-AT (differential drive) as slip and drift for the 3-AT robot are very high on the lab floor. Second, we changed the visual tags for landmarks and goods as we have previously been suffering from mixups in tag detection. Last but not least we extended the TreeMap SLAM algorithm[3] to provide uncertainty estimates. By exploiting covariances for position estimates from map and observation we are able to detect movements more robustly which increases the overall map quality too.

[3] As provided at http://www.openslam.org/TreeMap.html

8 Conclusion

In this paper, we propose an approach to process detection based on a specification of processes as temporal logic formulas in LTL. We demonstrate the applicability of our approach by an evaluation with real sensory data from a mobile robot. In our case study of warehouse logistics, the observations of a robot can be queried for process occurrences using an abstract process description. This allows a domain expert to obtain valuable information.

The evaluation demonstrates usefulness of the LTL-based approach to process description and recognition. With LTL one takes a declarative approach that is accessible to any domain expert as the declarative formulas abstract from the details of underlying algorithms. The performance of the gold standard clearly demonstrates feasibility of the symbolic approach to process specification and recognition, confirming the first claim of this paper. The claim is further supported by the actual recognition rates of the autonomous process recognition. Let us now consider the second claim of this paper, namely that the declarative approach enables logic reasoning to supplement observations of the robot, sensibly filling in missing pieces of information. Indeed, the experimental setting in which no information about zone membership is available a priori resembles a chicken-and-egg problem: on the one hand, zones need to be known in order to identify processes. On the other hand, the processes need to be known to identify zones. Approaching process recognition as a model checking problem allows us to jointly recognize processes and zones using the well-defined semantics of answer set programming. Naturally, the less information is available the poorer the recognition rate. Figure 7 however shows that the declarative approach effectively counteracts the loss of information, showing only a small decline despite loss of zone information. This demonstrates a key benefit of a logic-based approach: the seamless integration of inference processes into the robot control architecture. Last but not least, the approach is sufficiently efficient to handle real-world data. Two factors are the essential contributors: the qualitative representation cuts down comprehensive experimental runs to few observations (see Table 1) and the ASP solver exhibits a low-degree polynomial growth of computing time.

In a real-world warehouse we expect the robot to only observe a relative small amount of processes occurring, as a robot's perception is limited in range. Nevertheless, an analysis of the overall warehouse processes is still possible if the processes and histories detected are prototypical for the overall warehouse. This requires a high degree of correct recognitions though. As our approach meets this requirement we are confident that the approach will also scale to a large setting.

While the focus of this paper was to present an LTL-based approach to process recognition and understanding,

we aim to extend this approach to a comprehensive LTL-based control strategy. An interesting next step is to automatically derive an observation strategy that generates a sensible surveillance behaviour. In particular, we aim to use temporal logic to incorporate so-called search control knowledge and perform high-level planning [2], i.e., we shift to active process detection in the sense of planning which places to observe in order gather most valuable information.

For some complex queries it would be helpful to address all knowledge gathered during observations, in particular information about goods we have observed before and which are included in the map, but which we are unable to perceive at the very moment. Currently, we take a conservative approach that only explicates knowledge that is certain. However, for such objects we still have a strong belief of their existence and position in the warehouse, but this belief can—according to the actual observation—not be validated. A possibility to include reasoning on such beliefs is to use a logic that provides a modal belief operator, such as the logic for BDI agents presented in [26]. Another source of information for more complex queries could be provided by an ontology, as shown in [25].

Acknowledgments This paper presents work carried out in the project R3-[Q-Shape] of the Transregional Collaborative Research Center SFB/TR 8 Spatial Cognition. Financial support by the German Research Foundation (DFG) is gratefully acknowledged. We like to thank Udo Frese for his valuable comments and his support in extending the TreeMap-algorithm. We also thank the anonymous reviewers for their helpful comments.

References

1. Antoniotti M, Mishra B (1995) Discrete event models + temporal logic = supervisory controller: automatic synthesis of locomotion controllers. In: Proceedings of the IEEE conference on robotics and automation (ICRA), Nagoya, vol 2, pp 1441–1446
2. Bacchus F, Kabanza F (2000) Using temporal logics to express search control knowledge for planning. Artif Intell 116(1–2):123–191
3. Balcan MF, Blum A (2010) A discriminative model for semi-supervised learning. J ACM (JACM) 57(3):1–46
4. Bauland M, Mundhenk M, Schneider T, Schnoor H, Schnoor I, Vollmer H (2011) The tractability of model checking for LTL: the good, the bad, and the ugly fragments. ACM Trans Comput Log 12(2):13
5. Bennewitz M, Burgard W, Cielniak G, Thrun S (2005) Learning motion patterns of people for compliant robot motion. Int J Robot Res (IJRR) 24(1):39–41
6. Cimatti A, Giunchiglia F, Giunchiglia E, Traverso P (1997) Planning via model checking: a decision procedure for AR. In: Steel S, Alami R (eds) European conference on planning (ECP), Lecture notes in computer science. Springer, Berlin, vol 1348, pp 130–142
7. Dianco A, Alfaro LD (1995) Model checking of probabilistic and nondeterministic systems. In: Foundations of software technology and theoretical computer science. Springer, LNCS, vol 1026, pp 499–513
8. Ding XC, Smith SL, Belta C, Rus D (2011) Ltl control in uncertain environments with probabilistic satisfaction guarantees. Technical report accompanying IFAC 2011 (math.OC, arXiv: 1104.1159v2)
9. Dubba K, Bhatt M, Dylla F, Cohn A, Hogg D (2011) Interleaved inductive–abductive reasoning for learning event-based activity models. In: Inductive logic programming (lecture notes in computer science), 21st international conference, ILP-2011, Windsor Great Park
10. Edelkamp S, Jabbar S (2006) Action planning for directed model checking of petri nets. Electron Notes Theor Comput Sci 149(2):3–18
11. Frese U (2004) An $O(\log n)$ algorithm for simultaneous localization and mapping of mobile robots in indoor environments. PhD thesis, University of Erlangen-Nürnberg, Bavaria
12. Gebser M, Kaufmann B, Neumann A, Schaub T (2007) Conflict-driven answer set solving. In: Veloso M (ed) Proceedings of the 20th international joint conference on artificial intelligence (IJCAI'07), AAAI Press/MIT Press, pp 386–392
13. Gebser M, Kaminski R, König A, Schaub T (2011) Advances in gringo series 3. In: Delgrande J, Faber W (eds) Proceedings of the 11th international conference on logic programming and nonmonotonic reasoning (LPNMR'11), lecture notes in artificial intelligence, vol 6645, Springer, Berlin, pp 345–351
14. Hildebrandt T, Frommberger L, Wolter D, Zabel C, Freksa C, Scholz-Reiter B (2010) Towards optimization of manufacturing systems using autonomous robotic observers. In: Proceedings of the 7th CIRP international conference on intelligent computation in manufacturing engineering (ICME)
15. Kloetzer M, Belta C (2006) LTL planning for groups of robots. In: Proceedings of the IEEE international conference on networking, sensing and control (ICNSC), Fort Lauderdale, pp 578–583
16. Kloetzer M, Belta C (2010) Automatic deployment of distributed teams of robots from temporal logic motion specifications. IEEE Trans Robot 26(1):48–61
17. Kress-Gazit H, Wongpiromsarn T, Topcu U (2011) Correct, reactive robot control from abstraction and temporal logic specifications. Spec Issue IEEE Robot Autom Mag Form Methods Robot Autom 18(3):65–74
18. Kreutzmann A, Colonius I, Frommberger L, Dylla F, Freksa C, Wolter D (2011) On process recognition by logical inference. In: Proceedings of the 5th European conference on mobile robots (ECMR), Orebro, pp 7–12
19. Lahijanian M, Andersson SB, Belta C (2011) Temporal logic motion planning and control with probabilistic satisfaction guarantees. IEEE Trans Robot 99:1–14
20. Liao L, Patterson DJ, Fox D, Kautz H (2007) Learning and inferring transportation routines. Artif Intell 171(5–6):311–331
21. Lichtenstein O, Pnueli A (1985) Checking that finite state concurrent programs satisfy their linear specification. In: Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on principles of programming languages (POPL '85), ACM, New York, pp 97–107
22. Lifschitz V (1996) Foundations of logic programming. In: Brewka G (ed) Principles of knowledge representation. CSLI Publications, Stanford, pp 69–128
23. Lifschitz V (2002) Answer set programming and plan generation. Artif Intell 138:39–54
24. Mastrogiovanni F, Scalmato A, Sgorbissa A, Zaccaria R (2009a) On situation specification in context aware robotics applications. In: Proceedings of the 4th European conference on mobile robots (ECMR), Mlini/Dubrovnik, Croatia, pp 265–270
25. Mastrogiovanni F, Sgorbissa A, Zaccaria R (2009b) Context assessment strategies for ubiquitous robots. In: IEEE international conference on robotics and automation (ICRA) , pp 2717–2722
26. Meyer JJ, Veltman F (2007) Intelligent agents and common sense reasoning. In: Blackburn P, Benthem JV, Wolter F (eds) Handbook

of modal logic, studies in logic and practical reasoning, vol 3. Elsevier, New York, pp 991–1029 (chapter 18)
27. Morgenstern A, Schneider K (2011) Program sketching via CTL* model checking. In: Groce A, Musuvathi M (eds) Model checking software (SPIN), vol 6823. Springer, LNCS, Snowbird, pp 126–143
28. Nilsson NJ (1984) Shakey the robot. Technical report 323, AI Center, SRI international, Menlo Park
29. Pnueli A (1977) The temporal logic of programs. In: Proceedings of the 18th annual symposium on foundations of computer science (FOCS), pp 46–57
30. Schnoebelen P (2003) The complexity of temporal logic model checking. In: Balbiani P, Suzuki NY, Wolter F, Zakharyaschev M (eds) Selected papers from the 4th workshop on advances in modal logics (AiML'02). King's College Publication, Toulouse, pp 393–436
31. Sistla AP, Clarke EM (1985) The complexity of propositional linear temporal logics. J Assoc Comput Mach 32(3):733–749
32. Smith SL, Tumová J, Belta C, Rus D (2010) Optimal path planning under temporal logic constraints. In: Proceeding of the IEEE/RSJ international conference on intelligent robots and systems (IROS), Taipei, pp 3288–3293
33. Ten Hompel M, Schmidt T (2010) Management of warehouse systems. Springer, Berlin, pp 13–63 (chapter 2)
34. Wongpiromsarn T, Topcu U, Murray R (2009) Receding horizon temporal logic planning for dynamical systems. In: Proceedings of the 48th IEEE Conference on decision and control, 2009 held jointly with the 28th Chinese control conference. CDC/CCC 2009, pp 5997–6004
35. Wongpiromsarn T, Topcu U, Murray RM (2010) Receding horizon control for temporal logic specifications. In: Proceedings of the 13th ACM international conference on hybrid systems: computation and control, ACM, New York, HSCC'10, pp 101–110
36. Yang Q (2009) Activity recognition: linking low-level sensors to high level intelligence. In: Proceedings of the 21st international joint conference on artificial intelligence (IJCAI), Pasadena, pp 20–25