LiMa: Sequential Lifted Marginal Filtering on Multiset State Descriptions

Max Schröder, Stefan Lüdtke, Sebastian Bader,
Frank Krüger and Thomas Kirste

Mobile Multimedia Information Systems Group, Institute of Computer Science
University of Rostock, 18051 Rostock, Germany
{max.schroeder, stefan.luedtke2, sebastian.bader, frank.krueger2,
thomas.kirste}@uni-rostock.de

Abstract. Maintaining the a-posteriori distribution of categorical states given a sequence of noisy and ambiguous observations, e. g. sensor data, can lead to situations where one observation can correspond to a large number of different states. We call these states *symmetrical* as they cannot be distinguished given the observation. Considering each of them during the inference is computationally infeasible, even for small scenarios. However, the number of situations (called *hypotheses*) can be reduced by abstracting from particular ones and representing all symmetrical in a single abstract state. We propose a novel Bayesian Filtering algorithm that performs this abstraction. The algorithm that we call *Lifted Marginal Filtering* (LiMa) is inspired by Lifted Inference and combines techniques known from Computational State Space Models and Multiset Rewriting Systems to perform efficient sequential inference on a parametric multiset state description. We demonstrate that our approach is working by comparing LiMa with conventional filtering.

1 Introduction

Maintaining the a-posteriori distribution of categorical states given a sequence of noisy and ambiguous observations, e. g. sensor data, can lead to situations where one observation can correspond to a large number of different states. For example, when tracking persons based on anonymous presence sensors, we do not know which concrete person corresponds to which observation (track) [9, 25, 11]. We call such persons (more general entities) *observation equivalent*, i. e. they cannot be distinguished, based on the current observation. Thus, the number of states that need to be considered can grow very large, even for small scenarios. For example, when tracking the location and movement of 6 persons in 10 rooms, there are already 10^6 possible states. Even though this is a theoretical number of states, also the states that actually need to be tracked and cannot be precluded given the observation (also called *hypotheses*) is large. Thus, inference quickly becomes infeasible for real-world sized domains due to the combinatorial explosion with respect to the number of hypotheses that need to be tracked.

Several approaches exists that try to exploit such *symmetries.* Approaches that abstract from the identity of the entities [11, 16] cannot be used, because *identifying observations* might reveal the correspondence between some of the tracks and some of the identities. Additionally, in order to answer application specific questions, we might need the identity of entities. Such scenarios require an inference algorithm that can represent observation equivalent entities as a group, thus compactly representing states that are different but cannot be distinguished given the sensor data. However, this approach must also be able to break symmetries, i. e. split groups, when indicated by observations. Recently, Lifted Inference [18, 12, 23] showed that inference in graphical models can be performed on a first-order level, by reasoning over equivalent random variables as a group. However, none of these approaches allows to *recursively* compute the a-posteriori distribution, which is necessary when the complete observation sequence is not known in advance such as when using real world sensor data.

We propose a novel filtering algorithm that can compactly represent symmetrical states. It employs a multiset state representation that allows to group observation equivalent entities. This abstract state representation is embedded in the Bayesian Filtering framework in order to recursively compute the a-posteriori state distribution. For this purpose, the next belief state according to a transition model is *predicted* followed by an *update* of the probabilities according to an observation model that takes the current observation into account. This allows us to reason over them as a group, leading to a more compact belief state and a much more efficient filtering algorithm. To exemplify our approach, we will use the following office scenario [22] as a running example throughout the paper:

Example 1 [Office scenario] Up to six agents are in an office building with five rooms and a hall connecting the rooms. There are also two coffee machines and ten coffee capsules in a storage. Agents can walk between the rooms and take a capsule from the storage, respectively, replenish the coffee machine. If a capsule is inserted in the coffee machine, agents can take a coffee. All rooms contain presence sensors that detect if at least one agent is present. Our goal is to track the current state (positions of agents, items the agents are carrying as well as the number of capsules left and the state of the coffee machine etc.) based on sequences of presence sensor data.

The agents in this scenario are observation equivalent, as they cannot be distinguished given the presence sensor data. Although this scenario is a specific instance, the underlying problem is more general and can be found in many similar scenarios involving multiple observation equivalent entities acting in parallel. Note that this scenario requires modeling of entities along with their properties and thus cannot be solved by considering the number of entities only.

In Section 2, we introduce basic concepts that lay the foundations for our novel inference approach. The inference approach itself is described in Section 3. Section 4 evaluates the inference mechanism and Section 5 presents connections to other methods and related work. We finish this paper with our conclusion and a description of our future work in Section 6.

2 Preliminaries

In the following, we will give a brief overview of two concepts our approach is based on. Computational State Space Models allow Bayesian Filtering in a state space described by precondition-effect actions. Multiset Rewriting Systems offer a formalism for compactly representing states with multiple equivalent entities.

Computational State Space Models (CSSMs) allow the knowledge-based construction of state spaces for Bayesian Filtering. They are for instance used for human behavior and goal recognition [2, 19]. The transition model is described by a computable function by means of preconditions and effects. This allows the compact representation of potentially infinite state spaces by avoiding explicit state enumeration. Standard methods for Bayesian Filtering (e. g. Particle Filtering) is used to estimate the most likely state sequence.

CSSMs allow to handle large, even infinite, state spaces [17]. However, CSSMs perform inference in grounded state spaces (i. e. concrete values are assigned to all state variables). For representing observation equivalent states, this means the approach needs to track all of the different observation equivalent states individually, leading to a combinatorial explosion.

Multiset Rewriting Systems (MRSs) are an established formalism for modeling systems with many equal objects. They are for instance used to model chemical reactions happening in a solution [4] or cell interactions [5]. The state of such a system is described as a multiset of *entities*, where each entity is an instance of one of finitely many *species*. The reactions between entities are modeled as *multiset rewriting rules* that have preconditions (a multiset of entities that are consumed by the reaction) and effects (a multiset of entities that are created by the reaction). Under the probabilistic maximally parallel semantics [3], a maximal set of applicable rules (a compound rule) is applied in parallel. Each rule is assigned a *rate*, which defines the probability of a compound rule.

We are interested in MRS because they allow an abstract representation of states with multiple, equivalent entities. However, Bayesian Filtering algorithms for MRSs that incorporate observations have not yet been devised.

3 LiMa: Lifted Marginal Filtering

In the following, we present our approach that performs Bayesian Filtering using a multiset-based state representation. Our concept of Bayesian Filtering in state spaces described by precondition-effect actions is based on CSSMs (cf. Section 2). The state space representation is inspired by MRSs, which enable a compact representation of multiple equivalent states.

This section aims at giving a comprehensive overview of our *Lifted Marginal Filtering* approach. The next section is concerned with the question how states can be formalized in an abstract manner to represent multiple observation equivalent situations (Section 3.1). Section 3.2 extends this abstract representation to be capable of expressing uncertainty. The efficient manipulation of this uncertain abstract representation regarding a model of the system's dynamics is introduced in Section 3.3. Section 3.4 describes how observations can be taken into account to perform a complete Bayesian Filtering cycle. As Bayesian Filtering is used to answer application specific questions, in particular questions about the activity of the entities, we discuss how this can be performed in LiMa in Section 3.5.

3.1 Abstract State Description

Similar to MRS, we model a state as a multiset of entities with certain properties, e. g. objects or persons, that are part of a situation. Such entities often have many properties in common, but some properties with different values. For example, two persons may both be at the same location and both holding nothing in their hands, but having different names. In MRSs, these two persons are considered a different species. Thus, inference in MRS with many entities that are not exactly equal leads to a combinatorial explosion in the number of species. This combinatorial explosion can be avoided by extending the multiset representation to be able to group entities that are similar, but not equal.

For this purpose, our state space representation separates the *structure* of the entities from the actual property values of these entities, allowing us to group entities with similar structure, but different property values. While the number as well as the structure of entities is maintained in what we call a *state formula*, the possibly uncertain property values are maintained in the *context*. The context contains *representations* of densities[1] encoding the uncertainty respectively certainty over the entity properties. It is connected to the structure via *density labels*. Our inference algorithm manipulates the structure (the state formula), as well as these representations. Below, we introduce the concepts of entities, state formulae and contexts in detail including examples referring to the office scenario (Section 1).

An entity is a finite map of property names (called *slots*) to *density labels*. These density labels are used as a "name" for the possibly uncertain property values in the form of density representations that are later defined in the context.

Example 2 Let $\mathcal{E}$ be an entity that models agents with three slots $\mathsf{Location}$, Holds and Name, with $\mathcal{E}(\mathsf{Location}) = LHall$, $\mathcal{E}(\mathsf{Holds}) = LNil$ and $\mathcal{E}(\mathsf{Name}) = LNames$. We represent the entity as $\mathcal{E} = \langle \mathsf{Location}\colon LHall, \mathsf{Holds}\colon LNil, \mathsf{Name}\colon LNames \rangle$.

Multiple entities involved within a scenario are encoded using multisets[2] such that multiple similar entities are grouped together: Let $\mathscr{E} := \{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_n\}$ be entities and let $i_1, \ldots, i_n$ be natural numbers. A state formula over $\mathscr{E}$ is defined as a multiset over $\mathscr{E}$. We use $[\![\, i_1\mathcal{E}_1, i_2\mathcal{E}_2, \ldots, i_n\mathcal{E}_n \,]\!]$ to represent multisets of entities with corresponding cardinalities.

[1] We use the term *density* to refer to densities over continuous domains as well as probability distributions over finite domains.

[2] A multiset over some set S is defined as a partial map from S to $\mathbb{N}$. We use $[\![\, n_1 s_1, n_2 s_s, n_3 s_3 \,]\!]$ to denote the multiset containing s_1, s_2 and s_3 with the corresponding cardinalities. We use $\mathcal{M}(S)$ to refer to the set of all multisets over S.

Example 3 The situation in which 5 agents are located in the hall and another agent is located in room A can be represented as a state formula ϕ as follows:

$$\begin{aligned}\phi = [\![\, & 5\langle \textsf{Location: } \mathit{LHall}, \textsf{Holds: } \mathit{LNil}, \textsf{Name: } \mathit{LNames}\rangle, \\ & 1\langle \textsf{Location: } \mathit{LRoomA}, \textsf{Holds: } \mathit{LNil}, \textsf{Name: } \mathit{LNames}\rangle \,]\!]\end{aligned}$$

After modeling the structure as well as the number of entities, the actual property values are to be defined. The state formula is connected to the corresponding context that encodes the actual property values via density labels: A *context* is a finite map from density labels to density representations.

Below, we assume that, given a representation r of a density function d, there exists an algorithm Split, which accepts r and a value v as input and returns a representation r' of a density d' that is the result of removing v from d. We furthermore assume an algorithm Likelihood, which accepts r and a value v as input and returns the likelihood of v with respect to d. For example, let $r = \mathcal{U}(a, b, c)$ represent a finite urn containing the items a, b and c, then $\textsc{Split}(r, a)$ must return the representation of an urn containing b and c, and $\textsc{Likelihood}(r, v)$ will give 1/3 for each item.

A context γ is called *valid* wrt. a given state formula ϕ , if and only if for all density labels occurring in ϕ there exists a density representation in γ and all density representations occurring in γ are referenced within ϕ. Furthermore, every density function d encoded in the context γ must be able to be split using Split at least as many times as the sum of the cardinalities of the entities referencing this density. As an example, a context that connects *LNames* to an urn with three values only is not valid for the state formula as in Example 3, because the density is referenced six times.

In the example below, we use $\delta(x)$ to represent a density function which is non-zero for x only (i. e. we use $\delta(x)$ to refer to Dirac delta for continuous domains and the Kronecker delta for finite domains), we call $\delta(x)$ to be a *singleton distribution*. Note that singleton distributions cannot be split according to Split and instead returns the same distribution. $\mathcal{U}$ is used to represent a finite urn as described above. Note that when using the $\delta(x)$ density, we might draw x multiple times compared to $\mathcal{U}(x)$.

Example 4 Let two contexts γ_1 and γ_2 be defined as follows and let ϕ be the state formula as in Example 3, then γ_1 is not valid whereas γ_2 is valid for ϕ .

$$\begin{aligned}\gamma_1 &= \{\, \mathit{LHall} \mapsto \delta(\mathit{hall}), \mathit{LNil} \mapsto \delta(\mathit{nil}), \mathit{LNames} \mapsto \mathcal{U}(\mathrm{a}, \ldots, \mathrm{f}) \,\} \\ \gamma_2 &= \gamma_1 \cup \{\, \mathit{LRoomA} \mapsto \delta(\mathit{roomA}) \,\}.\end{aligned}$$

Below we assume all contexts to be valid contexts. A pair $\phi\gamma$ of state formula ϕ and valid context γ is called a *lifted state*. Note that by using this representation we assume all densities in the context to be independent from each other.

Example 5 The two situations (a) six agents are in the hall, and (b) five are in the hall and the sixth is in room A can be modeled as lifted state s_1 and s_2

as follows:

$$\begin{aligned} s_1 &= [\![\, 6\langle \textsf{Location: } LHall, \textsf{Holds: } LNil, \textsf{Name: } LNames\rangle \,]\!]\gamma_1 \\ s_2 &= [\![\, 5\langle \textsf{Location: } LHall, \textsf{Holds: } LNil, \textsf{Name: } LNames\rangle, \\ &\qquad 1\langle \textsf{Location: } LRoomA, \textsf{Holds: } LNil, \textsf{Name: } LNames\rangle \,]\!]\gamma_2 \end{aligned}$$

Note that representing s_2 in conventional grounded approaches would require to track at least six different hypotheses, namely agent a, b, c, d, e, or f being in room A. In our formalization, however, these situations are encoded using the single hypothesis s_2.

This formalism allows to represent multiple observation equivalent states as a single lifted state. Note that additional to the connection of MRS this representation employs Rao-Blackwellization: Some aspects of the state are described explicitly (via the state formula), while some aspects have a parametric representation (via the context).

3.2 Handling Uncertainty over Lifted States

Lifted states enable the modeling of groups of situations, i. e. groups of conventional states. However, as there are several sources of noise (observations, non-deterministic actions, ...), we will consider not just a single lifted state, but a probability distribution over lifted states as CSSMs maintain a probability distribution over grounded states. We call this probability distribution *lifted belief state*.

Before introducing it in detail, we define the concepts of *grounded states*: A ground state is a lifted state if and only if its context consists of singleton distributions only. Ground states correspond to the states used in conventional Bayesian Filtering: Each ground state represents a specific situation, while a lifted state in general represents a set of situations. We call this set of situations *state instances*, i. e. the state instances are the set of ground states that are subsumed under a lifted state. Note that this set is infinite if one of the underlying densities has an infinite domain.

A probability distribution over lifted states, called *lifted belief state*, thus specifies a probability distribution over sets of grounded states. We will use *lifted belief states* to represent the current set of hypotheses while tracking activities based on noisy observations.

Example 6 Let $S = \{s_1, s_2\}$ with s_1 and s_2 be as given in Example 5. Let $b(s_1) = 0.75$ and $b(s_2) = 0.25$. Then b is a belief state over S. Below we will use the following notation to describe belief states: $b = \{0.75 \times s_1, 0.25 \times s_2\}$. b describes the situations in which with probability 0.75 all six agents are in the hall, and with probability 0.25 one of them is in room A.

3.3 Abstract State Dynamics

After describing how states can be modeled in an abstract manner (cf. lifted states) and how uncertainty about the actual state can be represented (cf. lifted belief state), in this section, we describe how the dynamics of the system is modeled and how the representation is manipulated efficiently. This corresponds to the predict step of Bayesian Filtering. We call the function that maps a lifted belief state to a successor lifted belief state the *transition model*.

We use precondition-effect actions to model the dynamics of the system. The transition model is a parallel execution of multiple actions that are all applicable in the current state, similar to MRSs.

An action maps a set of entities satisfying the precondition (specific slots and slot values) to a new set of entities. These new entities are obtained by removing entities, by creating new ones, or by modifying entities (updating slot values, removing slots or adding slots). Before defining actions, we introduce a notion of slot and entity constraints:

Slot constraints check if a single value satisfies a condition. That is, slot constraints are Boolean functions of slot values indicating whether the condition is satisfied. We denote such functions as $sc := \lambda\, v \mapsto v \equiv v_{\text{test}}$. v is the property value to be evaluated and $v \equiv v_{\text{test}}$ is a Boolean expression of the property value, e. g. a test for (in)equality wrt. a given value, or set membership of simply the constant function *true* and *false*.

Multiple slot constraints then are combined in an *entity constraint* that maps slot names on slot constraints. Thus, an entity satisfies an entity constraint if: (1) the entity possesses all slots that are connected to a slot constraint, and (2) all slot constraints are satisfied. Note that considering the corresponding context may be necessary to decide on the satisfaction of slot constraints.

Example 7 Let $sc_h := (\lambda\, v \mapsto v \equiv \mathit{hall})$ be the slot constraint testing if the given slot value v is identical to the value '*hall*', $sc_\top := (\lambda\, v \mapsto \top)$ be the slot constraint used to ensure the presence of a given slot. Then $ec_1 = \{\mathsf{Location} \mapsto sc_h\}$ is an entity constraint, satisfied by all entities and corresponding contexts with a slot $\mathsf{Location}$ whose value is *hall*, and $ec_2 = \{\mathsf{Name} \mapsto sc_\top\}$ is satisfied by all entities which posses the slot Name.

As mentioned above, actions can modify the set of entities. A function transforming an entity into a new one is called *entity update function*. Possible entity update functions include the addition of new slots, the update of slot values or the removal of slots. These operations always include the modification of the corresponding context γ. However, for ease of understanding, we omit to mention that the context has always to be updated accordingly. I. e. given an entity $\mathcal{E}$, we use $\mathcal{E}\{\mathsf{s} \mapsto v\}$ to refer to the entity which results by setting the slot s to the value v (i.e., addition or update of s), and we use $\mathcal{E}\{-\mathsf{s}\}$ to refer to the entity obtained by removing slot s.

The effects of an action are specified by an *effect function* mapping a tuple of entities to a multiset of entities. This multiset can contain new entities, and entities resulting from performing entity update functions on the original entities. An *action schema* is the specification of an action, consisting of (1) a name, (2) a sequence of entity constraints π (preconditions), and (3) an effect function ϵ.

Example 8 For ec_1 as in Ex. 7, the schema ('H2A', $[ec_1]$, $(\mathcal{E}) \mapsto [\![\, 1\mathcal{E}\{\mathsf{Location} \mapsto \mathit{RoomA}\} \,]\!]$) captures the movement from the hall to room A.

An action is applicable in a given lifted state if and only if the state contains entities which satisfy the actions preconditions. However, if a lifted state contains an entity possessing all slots required by the entity constraint but with a non-singleton distribution in one of these slots, we cannot decide whether the entity satisfies the precondition. In this case, we split the corresponding lifted state into two lifted states: One where the precondition is satisfied, and one which contains all other grounded states that are instances of the original lifted states. Note the similarity to splitting in Lifted Inference [18]. These splits also involve the modification of the context as the densities encoding the uncertainty regarding the preconditions will be split (using Split) into two densities to remove the uncertainty. This does not necessarily require a complete grounding of the state, but only as far as needed to decide on the preconditions.

Example 9 Let act = ('H2A', $\pi_{act}, (\mathcal{E}) \mapsto [\![\, 1\mathcal{E}\{\mathsf{Location} \mapsto \mathit{RoomA}\} \,]\!]$) be an action schema, and $\pi_{act} = [\{\mathsf{Location} \mapsto sc_h, \mathsf{Name} \mapsto sc_a\}]$ be the corresponding precondition. Let $sc_h := (\lambda\, v \mapsto v \equiv \mathit{hall})$ and $sc_a = (\lambda\, v \mapsto v \equiv a)$ be the corresponding slot constraints. Then, act encodes the move action from the hall to room A performed by the agent named a. Considering the lifted state s_2 as in Example 5, there is no entity that already satisfies the action's preconditions. However, there is an entity in s_2 that is more general so that a modified version of this entity would satisfy the preconditions π_{act}. Thus, the lifted state can be split according to this entity on slot Name into the two lifted states: (1) a is at room A and the other agents are in the hall (satisfying the preconditions), and (2) a is at the hall, one of the other agents is in room A and the remaining are in the hall, too (not satisfying the preconditions). These splits include the modification of the context: the urn representation $\mathcal{U}(\mathrm{a}, \ldots, \mathrm{f})$ will be converted into $\mathcal{U}(\mathrm{b}, \ldots, \mathrm{f})$ and another density representation $\delta(a)$ will be inserted.

Splits, thus, can be used to ensure satisfaction of preconditions of an action schema. An action schema a together with a sequence of entities e satisfying the precondition is called an *action instance.* The entities in e are consumed while applying the action and replaced by the effect. I.e., the resulting state can *in principle* be computed by $s' = s \setminus e \cup \epsilon(e)$. Unfortunately, this would require the state s to be grounded. As described below it is also possible to compute the resulting state in a lifted manner. Before, we introduce the concept of maximal compound actions that encode the idea of maximally parallel actions in MRSs: A multiset of action instances is called *maximal compound action* (short: compound action) with respect to a lifted state if no further action instance can be added to the set so that the compound action can still be applied in the lifted state. Note that a compound action is applicable only if there is no entity referenced in two action instances.

Given a state formula ϕ, we can compute the successor states as follows:

1. Compute the set of maximal compound actions C

2. For each $c \in C$: (a) Compute the resulting splits, and (b) Compute the successor state s_c,
3. Merge the resulting successor states.

Predicting the successor states by a set of maximal compound actions might result in a set of lifted states that can be merged or pruned to reduce the number of hypotheses. A simple form of merging is summing probabilities of equal lifted states, as there might be multiple compound actions resulting in the same lifted state. Furthermore, multiple similar states can be merged by combining their entities so that the corresponding slot values are joined (exact or approximate). However, in this paper, we only perform simple merging by summing probabilities of equal lifted states.

3.4 Observation Model

In the previous section, we described state transitions based on actions. This corresponds to the *predict* step in Bayesian Filtering. In this section, we describe how the *update* step is realized in LiMa. This means, we want to manipulate the belief state, by use of an observation.

An observation is simply a condition on a property value, similar to a precondition of an action. The observation model OM takes a lifted belief state and the current observation to calculate a list of new states with updated probabilities for every lifted state of the belief state. I. e., the probabilities of the lifted states were weighted according to the current observation. For this purpose, the observation model splits each lifted state in the belief state on the observation and keeps only those lifted states that are consistent with the observation. The probabilities are then normalized to get a new valid belief state. Note that this procedure can easily be used for uncertain observations.

Example 10 For s_1 as in Example 5, let b_1 be a belief state with $b_1(s_1) = 1$. Observing $o = [\{\mathsf{Location} \mapsto sc_h, \mathsf{Name} \mapsto sc_a\}]$ with sc_h and sc_a as in Example 9, we get $OM(o, s_1) = \{1 \times s_1'\}$ with

$$\begin{aligned} s_1' = \llbracket\, & 5\langle \mathsf{Location}\colon \mathit{LHall}, \mathsf{Holds}\colon \mathit{LNil}, \mathsf{Name}\colon \mathit{LNames}' \rangle \\ & 1\langle \mathsf{Location}\colon \mathit{LHall}, \mathsf{Holds}\colon \mathit{LNil}, \mathsf{Name}\colon \mathit{LNameA} \rangle \,\rrbracket \\ & \{\, \mathit{LHall} \mapsto \delta(\mathit{hall}), \mathit{LNil} \mapsto \delta(\mathit{nil}), \mathit{LNames}' \mapsto \mathcal{U}(\mathrm{b}, \ldots, \mathrm{f}), \mathit{LNameA} \mapsto \delta(\mathrm{a}) \,\} \end{aligned}$$

After multiplying this with the probabilities in b_1 and normalizing it (both trivial in this example), we get $b_1' = \{1 \times s_1'\}$ as new belief state for observing o in b_1.

3.5 Reasoning over Lifted States

The predict and update steps (described in Section 3.3 and Section 3.4) together define a complete Bayesian Filtering cycle. As we aim at answering application specific questions during the inference, we need to be able to reason about the lifted belief state after every predict-update-cycle.

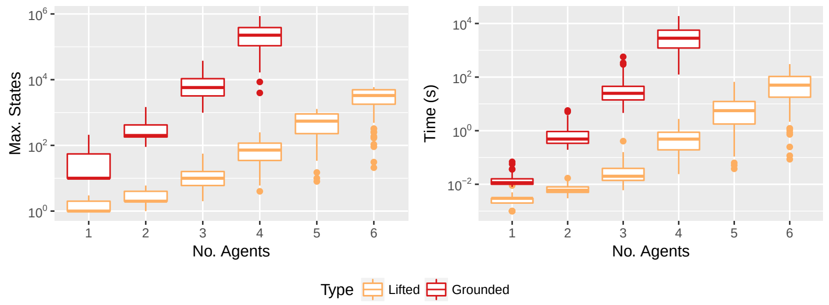

Fig. 1: Maximum number of (lifted) states during inference (left) and inference time (right) for grounded inference and Lifted Inference with LiMa. For the grounded inference, the scenarios with 5 and 6 agents could not be calculated due to the high computational effort. Note the log scale on the y axis.

Example 11 Considering the lifted belief state in Example 6, the question we want to answer is “Where is a?”. I. e. we want to calculate the distribution of values of the Location slot for entities with the Name being a. For this purpose, every lifted state in the lifted belief state needs to be evaluated against this question. In our example, both lifted states s_1 and s_2 contain only entities with slot Name mapping to the density $\mathcal{U}(\mathrm{a},\ldots,\mathrm{f})$. I. e. agent a is involved in any of those more general entities and thus the corresponding lifted states need to be split to decide on the position of a:

s_1) Splitting the entity in s_1 on Name with value a results in a single lifted state as the only possible Location for a is the *hall*. Thus, the probability of agent a being at the hall in s_1 is 1.0 resp. 0.75 (weighted by the probability of s_1).

s_2) There are 2 entities in s_2 with Name mapping to a density that includes agent a. A split on the name of agent a results in two possible lifted states: (a) a is at the hall, or (b) a is at room A. Whereas the first lifted state represents 5 grounded states, the second represents only one. Thus, a is at the *hall* with a probability of $\frac{5}{6}$ and at $roomA$ with a probability of $\frac{1}{6}$. These need to be weighted by the probability of the lifted state s_2.

Summing up the particular probability gives a probability that agent a is at the hall of 0.95833 and that the agent is at room A of 0.04167. Note that this split is for answering the application specific questions only. However, the un-split lifted states will be used for the further inference.

4 Evaluation

In the following, our approach is compared with a conventional Bayesian Filtering algorithm based on grounded states. As benchmark, we use the office dataset [21] that is described as office scenario in Section 1. It includes 720 observation sequences (120 for each number of agents between one and six) for which we perform activity recognition using both approaches.

Here, we have been particularly interested in the number of states considered during the inference task (i. e. the number of states in the belief state with non-zero support) as a measure of performance, as all approaches become infeasible if a very large number of hypotheses has to be considered. Furthermore, the relation between lifted and grounded states demonstrates the level of abstraction. For this office scenario, the number of agents is the factor determining the size of the state space. Therefore, we calculated the maximal number of states (i. e. hypotheses) maintained during Bayesian Filtering for each observation sequence. For LiMa, we counted the lifted states, and for the grounded approach the number of grounded states are considered. The results are shown in the left part of Figure 1. Note the log scale on the y axis.

The maximum number of states visited during Bayesian Filtering grows exponentially for both inference algorithms. However, for LiMa, the number of states is several orders of magnitude smaller than for the grounded state representation. Thus, LiMa successfully exploits observation equivalence by reducing the large number of grounded states to a much smaller number of lifted states. In fact, for the grounded state representation, Bayesian Filtering has been infeasible for problems with 5 or 6 agents due to the large number of states. Furthermore, considering the overall time necessary for each inference task, LiMa also performs several orders of magnitude faster than the grounded approach (see right part of Figure 1).

5 Related Work

There are several other approaches that perform efficient probabilistic inference or Bayesian Filtering on an abstract (e.g. logical) representation. A prominent approach concerned with inference in relational graphical models is known as Lifted Inference. The general idea is to exploit symmetries in the model, e.g. in cases where many objects with similar properties and relationships are present. We refer to [13, 15] for a more thorough overview. Opposed to LiMa, these methods do not explicitly support sequential inference in dynamic domains, i.e. Bayesian Filtering consisting of a predict-update cycle. The approach presented in [1] efficiently evaluates multiple Lifted Inference queries on the same network, but is not concerned with *dynamic* models, where random variables depends on random variables from previous time slices. Lifted Inference algorithms for dynamic models have also been devised [10], but this approach lacks an efficient way to preserve the lifted representation over time. Furthermore, it performs *approximate* inference, while LiMa is exact.

Ideas from Lifted Inference have also been used in the Relational Kalman Filter [8, 7]. This approach is similar to LiMa in the sense that it performs lifted Bayesian Filtering. That is, a compact representation of the belief state is maintained by grouping equivalent variables, and reasoning over them is performed "in bulk“. However, the approach can only be used for gaussian linear models, like the standard Kalman filter.

First-Order Markov Decision Processes (FOMDPs) [6, 20] employ first-order logic to represent states of a Markov Decision Process. The task performed in these formalisms is *lifted planning*, i.e. obtaining an abstract policy (that is independent of specific domain objects), given a goal. The algorithmic ideas used in this context decision-theoretic regression) are different from Bayesian Filtering applied by LiMa. However, there is a certain relationship between Lifted Inference and FOMDPs that has recently been discussed in [14].

6 Conclusion and Future Work

In this work, we presented a modeling formalism for abstract states that encodes multiple grounded states in a Bayesian Filtering context. Our approach that we call *Lifted Marginal Filtering* (LiMa) combines ideas of Computational State Space Models (CSSMs) and Multiset Rewriting Systems (MRS) to overcome the combinatorial explosion in grounded inference approaches. Our abstraction is based on *observation equivalence*, i.e. we reason over groups of situations that cannot be distinguished given the observations. Such groups (*lifted states*) are represented as a multiset of structure descriptions (entities) along with a *context* that describes the corresponding (possibly uncertain) values that can be inserted into that structure in the form of density functions. The transition model of LiMa is represented by precondition-effect actions similar to CSSMs that are combined to *compound actions* representing a maximally parallel application of such simple actions which is similar in MRS. We showed that applying actions and observations may require *splitting* of lifted states as in Lifted Inference, and derived a Bayesian Filtering algorithm that is capable of this representation and computes prediction and update in the lifted domain. To answer application specific questions, we demonstrated how to reason over lifted states. We expect that in many scenarios, these answers can often be computed without completely grounding and thus exploiting the lifted representation.

For an office scenario that suffers from a combinatorial explosion in the state space size, we showed that the state space size as well as the inference time is several orders of magnitude smaller than for the corresponding grounded inference.

Our approach can be extended in several ways. We will investigate the definition of a smoothing and MAP algorithm for the state representation. Furthermore, we plan to model time-dependency of state transitions, similar to Hidden Semi-Markov Models. Approximation is another interesting aspect: In some domains, *identifying* observations may lead to many splits, so that the algorithm actually resorts to grounded inference. This problem has been addressed before in Lifted Inference [24] by grouping states that are only approximately equal. In our case, this corresponds to approximate merging, which we plan to investigate in the future. A further aspect is to investigate which continuous densities can be used in the context, i. e. for which densities appropriate splitting functions can be defined that result in a compact representation of the split densities.

References

1. Ahmadi, B., Kersting, K., Sanner, S.: Multi-evidence lifted message passing, with application to pagerank and the kalman filter. In: Proceedings-International Joint Conference on Artificial Intelligence. p. 1152 (2011)
2. Baker, C.L., Saxe, R., Tenenbaum, J.B.: Action understanding as inverse planning. Cognition 113(3), 329–349 (2009)
3. Barbuti, R., Levi, F., Milazzo, P., Scatena, G.: Maximally Parallel Probabilistic Semantics for Multiset Rewriting. Fundamenta Informaticae 112(1), 1–17 (2011)
4. Berry, G., Boudol, G.: The chemical abstract machine. In: POPL. pp. 81–94. ACM, San Francisco, USA (1990)
5. Bistarelli, S., Cervesato, I., Lenzini, G., Marangoni, R., Martinelli, F.: On representing biological systems through multiset rewriting. In: EUROCAST. pp. 415–426. Springer, Las Palmas de Gran Canaria, Spain (2003)
6. Boutilier, C., Reiter, R., Price, B.: Symbolic dynamic programming for first-order MDPs. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence. vol. 1, pp. 690–700 (2001)
7. Choi, J., Amir, E., Xu, T., Valocchi, A.J.: Learning Relational Kalman Filtering. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. pp. 2539–2546 (2015)
8. Choi, J., Hill, D.J., Amir, E.: Lifted Inference for Relational Continuous Models. In: Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence. pp. 126–134. UAI'10, AUAI Press (2010)
9. Fox, V., Hightower, J., Liao, L., Schulz, D., Borriello, G.: Bayesian filtering for location estimation. IEEE Pervasive Computing 2(3), 24–33 (Jul 2003)
10. Geier, T., Biundo, S.: Approximate online inference for dynamic markov logic networks. In: 23rd IEEE International Conference on Tools with Artificial Intelligence. pp. 764–768. IEEE (2011)
11. Huang, J., Guestrin, C., Jiang, X., Guibas, L.: Exploiting Probabilistic Independence for Permutations. In: AISTATS. pp. 248–255. Clearwater, USA (2009)
12. Kersting, K., Ahmadi, B., Natarajan, S.: Counting belief propagation. In: UAI. pp. 277–284. Montreal, Canada (2009)
13. Kersting, K.: Lifted Probabilistic Inference. In: ECAI 2012 - 20th European Conference on Artificial Intelligence. Frontiers in Artificial Intelligence and Applications, vol. 242. IOS Press (2012)
14. Khardon, R., Sanner, S.: Stochastic planning and lifted inference. arXiv preprint arXiv:1701.01048 (2017)
15. Kimmig, A., Mihalkova, L., Getoor, L.: Lifted graphical models: a survey. Machine Learning 99 (2015)
16. Kondor, R., Howard, A., Jebara, T.: Multi-object tracking with representations of the symmetric group. In: AISTATS. vol. 2, pp. 211–218 (2007)
17. Krüger, F., Nyolt, M., Yordanova, K., Hein, A., Kirste, T.: Computational State Space Models for Activity and Intention Recognition. A Feasibility Study. PLOS ONE 9(11), e109381 (Nov 2014)
18. Poole, D.: First-order probabilistic inference. In: IJCAI. pp. 985–991 (2003)
19. Ramírez, M., Geffner, H.: Goal Recognition over POMDPs: Inferring the Intention of a POMDP Agent. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence. pp. 2009–2014 (Jul 2011)
20. Sanner, S., Boutilier, C.: Practical solution techniques for first-order MDPs. Artificial Intelligence 173, 748–788 (2009)

21. Schröder, M., Lüdtke, S., Bader, S., Krüger, F., Kirste, T.: An office scenario dataset for benchmarking observation-equivalent entities (2016), http://dx.doi.org/10.18453/rosdok_id00000138
22. Schröder, M., Lüdtke, S., Bader, S., Krüger, F., Kirste, T.: Abstracting from Observation-equivalent Entities in Human Behavior Modeling. In: AAAI Workshop: Plan, Activity, and Intent Recognition (Feb 2017)
23. Van Den Broeck, G., Taghipour, N., Meert, W., Davis, J., De Raedt, L.: Lifted probabilistic inference by first-order knowledge compilation. In: IJCAI. pp. 2178–2185 (2011)
24. Venugopal, D., Gogate, V.: Evidence-based clustering for scalable inference in markov logic. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 258–273. Springer (2014)
25. Wilson, D.H., Atkeson, C.: Simultaneous Tracking and Activity Recognition (STAR) Using Many Anonymous, Binary Sensors. In: Pervasive, pp. 62–79. Springer (2005), http://dx.doi.org/10.1007/11428572_5