Mach Learn (2011) 82: 239–272
DOI 10.1007/s10994-010-5213-8

Stochastic relational processes: Efficient inference and applications

Ingo Thon · Niels Landwehr · Luc De Raedt

Received: 1 June 2009 / Accepted: 1 May 2010 / Published online: 25 September 2010

Abstract One of the goals of artificial intelligence is to develop agents that learn and act in complex environments. Realistic environments typically feature a variable number of objects, relations amongst them, and non-deterministic transition behavior. While standard probabilistic sequence models provide efficient inference and learning techniques for sequential data, they typically cannot fully capture the relational complexity. On the other hand, statistical relational learning techniques are often too inefficient to cope with complex sequential data. In this paper, we introduce a simple model that occupies an intermediate position in this expressiveness/efficiency trade-off. It is based on CP-logic (Causal Probabilistic Logic), an expressive probabilistic logic for modeling causality. However, by specializing CP-logic to represent a probability distribution over sequences of relational state descriptions and employing a Markov assumption, inference and learning become more tractable and effective. Specifically, we show how to solve part of the inference and learning problems directly at the first-order level, while transforming the remaining part into the problem of computing all satisfying assignments for a Boolean formula in a binary decision diagram.

We experimentally validate that the resulting technique is able to handle probabilistic relational domains with a substantial number of objects and relations.

Keywords Statistical relational learning · Stochastic relational process · Markov processes · Time series · CP-Logic

Editors: S.V.N. Vishwanathan, Samuel Kaski, Jennifer Neville, and Stefan Wrobel.

I. Thon (✉) · L. De Raedt
Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
e-mail: ingo.thon@cs.kuleuven.be

L. De Raedt
e-mail: luc.deraedt@cs.kuleuven.be

N. Landwehr
Department of Computer Science, University of Potsdam, August-Bebel-Str. 89, 14482 Potsdam, Germany
e-mail: landwehr@cs.uni-potsdam.de

1 Introduction

One of the current challenges in artificial intelligence is the modeling of dynamic environments that change due to actions and activities people or other agents take. As one example, consider a model of the activities of a cognitively impaired person (Pollack 2005). Such a model can be used to assist persons, using common patterns to generate reminders or detect potentially dangerous situations, and thus help to improve living conditions.

As another example and one on which we shall focus in this paper, consider a model of the environment in a *massively multiplayer online game* (MMOG). These are computer games that support thousands of players in complex, persistent, and dynamic virtual worlds. They form an ideal and realistic testbed for developing and evaluating artificial intelligence techniques and are also interesting in their own right (cf. also Laird and van Lent 2000). One challenge in such games is to build a dynamic probabilistic model of high-level player behavior, such as players joining or leaving alliances and concerted actions by players within one alliance. Such a model of human cooperative behavior can be useful in several ways. Analysis of in-game social networks is not only interesting from a sociological point of view but could also be used to visualize aspects of the gaming environment or give advice to inexperienced players (e.g., which alliance to join). More ambitiously, the model could be used to build computer-controlled players that mimic the cooperative behavior of human players, form alliances and jointly pursue goals that would be impossible to attain otherwise. Mastering these social aspects of the game will be crucial to building smart and challenging computer-controlled opponents, which are currently lacking in most MMOGs. Finally, the model could also serve to detect non-human players in today's MMOGs—accounts which are played by automatic scripts to give one player an unfair advantage, and are typically against game rules.

From a machine learning perspective, this type of domain poses three main challenges: (1) the world state descriptions are inherently relational, as the interaction between (groups of) agents is of central interest, (2) the transition behavior of the world is strongly stochastic, and (3) a relatively large number of objects and relations is needed to build meaningful models, as the defining element of environments such as MMOGs are interactions among *large* sets of agents. Thus, we need an approach that is both computationally efficient and able to represent complex relational state descriptions and stochastic world dynamics. In this setting, a relation state typically corresponds to a labeled (hyper)graph, and therefore the model can also be viewed as a stochastic model over sequences of graphs, cf. Fig. 8.

Artificial intelligence has already contributed a rich variety of different modeling approaches, for instance, Markov models (Rabiner 1989) and decision processes (Puterman 1994), dynamic Bayesian networks (Ghahramani 1997), STRIPS (Fikes and Nilsson 1995) and PPDDL (Younes and Littman 2004) (see also Thon et al. (2009) for a discussion of the relationship between CPT-L and PPDDL), statistical relational learning representations (Getoor and Taskar 2007), etc. Most of the existing approaches that support reasoning about uncertainty (that is, satisfy requirement (2) employ essentially propositional representations (for instance, dynamic Bayesian networks, Markov models, etc.). Thus, they are not able to represent complex relational worlds, and do not satisfy requirement (1). A class of models that integrates logical or relational representations with methods for reasoning about uncertainty (for instance, Markov Logic (Richardson and Domingos 2006), CP-logic (Vennekens et al. 2006), or Bayesian Logic Programs (Kersting and De Raedt 2007)) is considered within statistical relational learning (Getoor and Taskar 2007) and probabilistic inductive logic programming (De Raedt et al. 2008). However, inference and learning often cause significant computational problems in realistic applications, and hence, such methods do not satisfy requirement (3).

We want to alleviate this situation, by contributing a novel representation, called CPT-L (for **C**ausal **P**robabilistic**T**ime-**L**ogic), that occupies an intermediate position in this expressiveness/efficiency trade-off. A CPT-L model essentially defines a probability distribution over sequences of interpretations. Interpretations are relational state descriptions that are typically used in planning and many other applications of artificial intelligence. CPT-L can be considered a variation of CP-logic (Vennekens et al. 2006), a recent expressive logic for modeling causality. By focusing on the sequential aspect and deliberately avoiding the complications that arise when dealing with hidden variables, CPT-L is more restricted, but also more efficient to use than alternative formalisms within the artificial intelligence and statistical relational learning literature.

The present paper builds upon our recent work in this area (Thon et al. 2008), and extends this earlier work in several directions. As a first contribution, we generalize the model presented in Thon et al. (2008) to include the case that head elements are conjunctions of atoms rather than individual atoms. Second, we relax the strict Markov assumption employed in the original approach, thus allowing causal influences to stretch over several time steps. Third, we present partially lifted algorithms for inference and learning in CPT-L, and empirically show that they reduce the overall computational complexity of our approach substantially. Finally, we contribute a more detailed and comprehensive experimental evaluation of our approach.

The rest of the paper is organized as follows: Sect. 2 introduces the CPT-L framework; Sect. 3 addresses inference and parameter estimation; and Sect. 4 presents experimental results in several (artificial and real-world) domains. Finally, we discuss related work in Sect. 5, before concluding and touching upon future work in Sect. 6. Proofs of the main theorems are contained in the Appendix.

2 CPT-L

This section describes the CPT-L model. We begin with a brief review of CP-logic (Causal Probabilistic Logic), and then present the CPT-L model as a variant of the more general CP-logic framework.

2.1 CP-logic

Let us first introduce some terminology. A logical *atom* is an expression of the form $p(t_1, \ldots, t_n)$ where p/n is a *predicate symbol* and the t_i are *terms*. Terms are built up from constants, variables, and functor symbols. Constants are denoted in lower case (such as a), variables in upper case (such as Z), and functors by f/k where k is the arity of functor f. The set of all atoms is called a *language* $\mathcal{L}$. *Ground* expressions do not contain variables. Ground atoms will be called *facts*. A substitution θ is a mapping from variables to terms, and $b\theta$ is the atom obtained from b by replacing variables with terms according to θ. As an example, consider the substitution $\theta = \{Z/a\}$ that replaces variable Z with a, as in $b\theta = p(a)$ for $b = p(Z)$.

Complex world states can now be described in terms of *interpretations*. An interpretation I is a set of ground facts $\{a_1, \ldots, a_N\}$. These ground facts can represent objects in the current world state, their properties, and any relationship between objects. As an example, consider the representation of the state of a multiplayer game in terms of an interpretation as depicted in Fig. 1.

The semantics of our framework is based on CP-logic, a probabilistic first-order logic that defines probability distributions over interpretations (Vennekens et al. 2006). CP-logic

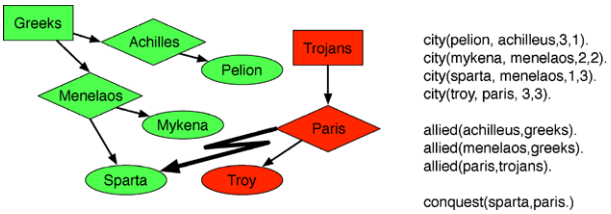

Fig. 1 Example for the state of a multiplayer game represented as a graph structure and, equivalently, as a logical interpretation. The *rectangles* in graphical representation refer to alliances, *diamonds* to players, and *ellipsis* to cities. The last two arguments of city in the logical representation refer to the location of the city

is closely related to other probabilistic logic programming systems, such as PRISM (Sato and Kameya 1997), ICL (Poole 1997), and ProbLog (De Raedt et al. 2007), that are based on Sato's distribution semantics (Sato 1995). However, CP-logic is more intuitive as a knowledge representation framework. The reason is that CP-logic has a strong focus on causality and constructive processes: an interpretation is incrementally constructed by a process that adds facts to the interpretation which are probabilistic *outcomes* of other already given facts (the *causes*). More formally, a model in CP-logic is defined as a set of (probabilistic) rules representing causes and outcomes:

Definition 1 A **CP-theory** is a set of rules of the form

$$r = (h_1 : p_1) \vee \cdots \vee (h_n : p_n) \longleftarrow b_1, \ldots, b_m$$

where the h_i are logical atoms, the b_i are literals (i.e., atoms or their negation) and $p_i \in [0, 1]$ probabilities s.t. $\sum_{i=1}^{n} p_i = 1$.

It will be convenient to refer to $b_1, \ldots, b_m$ as the $body(r)$ of the rule and to $(h_1 : p_1) \vee \cdots \vee (h_n : p_n)$ as the $head(r)$ of the rule. The body of rule is interpreted as a conjunction of literals. We shall also assume that the rules are range-restricted, that is, that all variables appearing in the head of the rule also appear in its body. The semantics of a CP-theory is given by the following probabilistic constructive process. Starting from the empty interpretation, at each step we consider all groundings $r\theta$ of rules r such that $body(r\theta)$ holds in the current interpretation. For each of these groundings, one of the grounded head elements $h_1\theta, \ldots, h_n\theta$ of r is chosen randomly according to the distribution given by $p_1, \ldots, p_n$. The chosen head element is then added to the current interpretation, and the process is repeated until no more new atoms can be derived. Note that each grounding of a rule can only contribute a single head element. This probabilistic process defines a generative probabilistic model over (functor-free) interpretations (Vennekens et al. 2006).

2.2 From CP-logic to CPT-L

CPT-L combines the semantics of CP-logic with that of (first-order) Markov processes. This corresponds to the assumption that for any sequence of interpretations there is an underlying generative process that constructs the next interpretation from the current one. More formally, a (discrete-time) *stochastic process* defines a distribution $P(X_1, \ldots, X_T)$ over a sequence of random variables $X_1, \ldots, X_T$ that characterize the state of the world at time $t = 1, \ldots, T$. We are interested in the case where X is a relational state description, that is, in *relational stochastic processes*. A relational stochastic process defines a distribution

$P(I_0, \ldots, I_T)$ over sequences of interpretations $I_0, \ldots, I_T$, where interpretation I_t describes the state of the world at time t. Thus, the random variable X_t describing the state of the process at time t is an interpretation, that is, a structured state. Such a process completely characterizes the (probabilistic) transition behavior of the world.

A stochastic process is called *Markov* if $P(X_{t+1} \mid X_t, \ldots, X_0) = P(X_{t+1} \mid X_t)$, and *stationary* if $P(X_{t+1} \mid X_t) = P(X_{t'+1} \mid X_{t'})$ for all t, t'. Stationary Markov processes are the simplest and most widely used class of stochastic processes, and thus are a natural starting point for developing simple models for relational stochastic processes. Additionally, we will assume full observability, meaning that the full state X can be directly observed in the data. While this is a restrictive assumption that will not be appropriate for all domains, it makes learning and inference in the resulting probabilistic model significantly easier.

The main idea behind CPT-L is to apply the causal probabilistic framework of CP-logic to stationary Markov processes, by assuming that the state of the world at time $t+1$ is a probabilistic outcome of the state of the world at time t. The constructive probabilistic process is thus unfolded over time, such that observed facts in interpretation I_t (probabilistically) cause other facts to be observed in I_{t+1}. In this setting, the first-order Markov assumption states that causal influences only stretch from I_t to I_{t+1}, but not further into the future. More formally, we define a *CPT-theory* as follows:

Definition 2 A **CPT-theory** is a set of rules of the form

$$r = (h_{1,1} \wedge \cdots \wedge h_{1,k_1} : p_1) \vee \cdots \vee (h_{n,1} \wedge \cdots \wedge h_{1,k_n} : p_n) \longleftarrow b_1, \ldots, b_m$$

where the $h_{i,j}$ are logical atoms, $p_i \in [0, 1]$ are probabilities s.t. $\sum_{i=1}^{n} p_i = 1$, and the b_l are literals (i.e., atoms or their negation).

A conjunction $h_{i,1} \wedge \cdots \wedge h_{i,k_i}$ in $head(r)$ will also be called a *head element*, and its probability p_i will be denoted by $P(h_{i,1} \wedge \cdots \wedge h_{i,k_i} \mid r)$. The meaning of a rule is that whenever $b_1\theta, \ldots, b_m\theta$ holds for a substitution θ in the current state I_t, exactly one head element $h_{i,1}\theta \wedge \cdots \wedge h_{i,k_i}\theta$ is chosen from $head(r)$ and all its conjuncts $h_{i,j}\theta$ are added to the next state I_{t+1}.

Note that in contrast to CP-logic, outcomes in CPT-L can be conjunctions of facts rather than individual facts. This is needed to represent causes with multiple outcomes in the next time step. In CP-logic, such multiple outcomes can be easily simulated using a set of rules of the form $h_{i,j} : 1 \longleftarrow h_i$ for $j = 1, \ldots, k_i$ that expand a single head element h_i into a conjunction $h_{i,1}, \ldots, h_{i,k_i}$. However, in CPT-L no new facts can be derived within one state I_t, thus such an expansion is not possible and conjunctions are needed to represent multiple outcomes.

Example 1 Consider the following CPT-theory for the *blocks world* domain:

$$r_1 = free(X) : 1.0 \longleftarrow free(X), \neg move(Y, X)$$

$$r_2 = on(X, Y) : 1.0 \longleftarrow on(X, Y), \neg move(X, Z), free(Z)$$

$$r_3 = (on(A, B) \wedge free(C) : 0.9) \vee (on(A, C) \wedge free(B) : 0.1)$$
$$\longleftarrow free(A), free(B), on(A, C), move(A, B).$$

The first two rules represent frame axioms, namely that a block stays free if no other block is moved upon it, and that blocks stay on each other unless they are moved. The third rule states that if we try to move block A on block C this succeeds with a probability of 0.9.

We now show how a CPT-theory defines a distribution over sequences $I_0, \ldots, I_T$ of relational interpretations. Let us first define the concept of an applicable rule r in an interpretation I_t. Consider a CPT rule $c_1 : p_1 \vee \cdots \vee c_n : p_n \longleftarrow b_1, \ldots, b_m$. Let θ denote a substitution that grounds the rule r, and let $r\theta$ denote the grounded rule. A rule r is applicable in I_t if and only if there exists a substitution θ such that $body(r)\theta = b_1\theta, \ldots, b_m\theta$ is true in I_t, denoted $I_t \models b_1\theta, \ldots, b_m\theta$. We will most often talk about ground rules that are applicable in an interpretation.

One of the main features of CPT-theories is that they are easily extended to include *background knowledge*. The background knowledge B can be any logic program, that is, a set of first-order clauses (cf. Bratko 1990). When working with background knowledge, the state I_t is represented by a set of facts and a ground rule is applicable in a state I_t if $b_1\theta, \ldots, b_m\theta$ can be inferred from I_t together with the background knowledge B. More formally, a ground rule is applicable if and only if $I_t \cup B \models b_1\theta, \ldots, b_m\theta$. To simplify the notation during the elaboration of our probabilistic semantics we shall largely ignore the use of background knowledge.

Given a CPT-Theory $\mathcal{T}$, the set of all applicable ground rules in state I_t will be denoted as $\mathbf{R}_t$. That is, $\mathbf{R}_t = \{r\theta \mid r \in \mathcal{T}, r\theta \text{ applicable in } I_t\}$. Each ground rule applicable in I_t will cause one of its grounded head elements to be selected, and the resulting atoms to become true in I_{t+1}. More formally, let $\mathbf{R}_t = \{r_1, \ldots, r_k\}$. A *selection* σ is a mapping from applicable ground rules $\mathbf{R}_t$ to head elements, associating each rule $r_i \in \mathbf{R}_t$ with one of its head elements $\sigma(r_i)$. Note that $\sigma(r_i)$ is a conjunction of ground atoms. The probability of σ is simply the product of the probabilities of selecting the respective head elements, that is,

$$P(\sigma) = \prod_{i=1}^{k} P(\sigma(r_i) \mid r_i) \tag{1}$$

where $P(\sigma(r_i) \mid r_i)$ is the probability associated with head element $\sigma(r_i)$ in the rule r_i.

A selection σ defines which head element is selected for every rule, and thus determines a successor interpretation I_{t+1}, that simply consists of all atoms appearing in selected head elements. More formally,

$$I_{t+1} = \bigwedge_{i=1}^{k} \sigma(r_i)$$

where, abusing notation, we have denoted an interpretation as a conjunction of atoms rather than a set of atoms. We shall say that σ *yields* I_{t+1} from I_t, denoted $I_t \xrightarrow{\sigma} I_{t+1}$, and define

$$P(I_{t+1} | I_t) = \sum_{\sigma : I_t \xrightarrow{\sigma} I_{t+1}} P(\sigma). \tag{2}$$

That is, the probability of a successor interpretation I_{t+1} given an interpretation I_t is computed by summing the probabilities of all selections yielding I_{t+1} from I_t. Note that $P(I_{t+1} \mid I_t) = 0$ if no selection yields I_{t+1}.

Example 2 Consider the theory

$$r_1 = p(X) : 0.2 \vee q(X) : 0.8 \longleftarrow q(X)$$
$$r_2 = p(a) : 0.5 \vee (q(b) \wedge q(c)) : 0.5 \longleftarrow \neg q(b)$$
$$r_3 = p(X) : 0.7 \vee nil : 0.3 \longleftarrow p(X).$$

Starting from $I_t = \{p(a)\}$ only the rules r_2 and r_3 are applicable, so $\mathbf{R}_t = \{r_2, r_3\{X/a\}\}$. The set of possible selections is $\Gamma = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ with

$$\sigma_1 = \{(r_2, p(a)), (r_3, p(a))\} \qquad \sigma_2 = \{(r_2, q(b) \wedge q(c)), (r_3, p(a))\}$$
$$\sigma_3 = \{(r_2, p(a)), (r_3, nil)\} \qquad \sigma_4 = \{(r_2, q(b) \wedge q(c)), (r_3, nil)\}.$$

The possible successor states I_{t+1} are therefore

$$I^1_{t+1} = \{p(a)\} \quad \text{with } P(I^1_{t+1} \mid I_t) = 0.5 \cdot 0.7 + 0.5 \cdot 0.3 = 0.5$$
$$I^2_{t+1} = \{q(b), q(c)\} \quad \text{with } P(I^2_{t+1} \mid I_t) = 0.5 \cdot 0.3 = 0.15$$
$$I^3_{t+1} = \{p(a), q(b), q(c)\} \quad \text{with } P(I^3_{t+1} \mid I_t) = 0.5 \cdot 0.7 = 0.35.$$

As for propositional Markov processes, the probability of a sequence $I_0, \ldots, I_T$ given an initial state I_0 is defined by

$$P(I_0, \ldots, I_T) = P(I_0) \prod_{t=0}^{T} P(I_{t+1} \mid I_t). \tag{3}$$

Intuitively, it is clear that this defines a distribution over all sequences of interpretations of length T as in the propositional case. More formally, inductive application of the product rule yields the following theorem:

Theorem 1 (Semantics of a CPT theory) *Given an initial state I_0, a CPT-theory defines a discrete-time stochastic process, and therefore for $T \in \mathbb{N}$ a distribution $P(I_0, \ldots, I_T)$ over sequences of interpretations of length T.*

2.3 Relaxing the Markov assumption

The CPT-L model described so far is based on a first-order Markov assumption (3). As for propositional Markov processes, it is straightforward to relax this assumption and allow higher-order dependencies such that

$$P(I_0, \ldots, I_T) = P(I_0) \prod_{t=0}^{T} P(I_{t+1} \mid I_{t-n+1}, \ldots, I_t)$$

where $n > 1$ is the model order. In particular, for $n = \infty$, we have a full-history model given by

$$P(I_0, \ldots, I_T) = P(I_0) \prod_{t=0}^{T} P(I_{t+1} \mid I_0, \ldots, I_t). \tag{4}$$

For propositional Markov processes, a naive representation of $P(I_{t+1} \mid I_{t-n+1}, \ldots, I_t)$ leads to a number of model parameters that is exponential in n. Thus, higher-order models typically require additional assumptions (as in Mixed Memory Markov Models (Saul and Jordan 1999)) and/or regularization to avoid overfitting and excessive computational complexity. However, in CPT-L we can easily take into account all previous interpretations when constructing a successor interpretation without a combinatorial explosion in model complexity. The idea is to extend rule conditions to match on all previous interpretations. This can be

realized by aggregating all previous interpretations $I_t, I_{t-1}, \ldots, I_0$ using fluents (facts extended with an additional argument for the timepoint), and then matching on the aggregated history. More formally, let $\mathcal{F}(I,t)$ denote the interpretation I where all facts have been extended by an additional argument t, as in $\mathcal{F}(I,0) = \{p(0,a), q(0,b)\}$ for $I = \{p(a), q(b)\}$. Now define the aggregated history as

$$I_{[0,t]} = \bigcup_{t'=0}^{t} \mathcal{F}(I_{t'}, t' - t).$$

CPT-L rules are still of the form

$$r = c_1 : p_1 \vee \cdots \vee c_n : p_n \longleftarrow b_1, \ldots, b_m$$

where the head elements c_i are conjunctions and the p_i probabilities as in Definition 2, but body literals b_i now match on the interpretation $I_{[0,t]}$. According to (4), we now need to construct a successor interpretation I_{t+1} given a history of interpretations $I_t, I_{t-1}, \ldots, I_0$, or, equivalently, giving the aggregated history $I_{[0,t]}$. In this new setting, a rule r is applicable given $I_t, I_{t-1}, \ldots, I_0$ if and only if there is a grounding θ such that $I_{[0,t]} \models b_1\theta, \ldots, b_m\theta$. As before, we probabilistically select for every applicable rule a grounded head element $c_i\theta$ and add its atoms to I_{t+1}.

Example 3 (Reconsider Example 2) In the new setting, rules r_1, r_2, r_3 can be written as

$$r_1 = p(X) : 0.2 \vee q(X) : 0.8 \longleftarrow q(0, X)$$
$$r_2 = p(a) : 0.5 \vee (q(b) \wedge q(c)) : 0.5 \longleftarrow \neg q(0, b)$$
$$r_3 = p(X) : 0.7 \vee nil : 0.3 \longleftarrow p(0, X).$$

Assume we are given a history $I_1 = \{p(a)\}, I_0 = \{q(b), q(c)\}$ and need to compute $P(I_2 \mid I_1, I_0)$. The joint interpretation is

$$I_{[0,1]} = \{p(0,a), q(-1,b), q(-1,c)\}.$$

The possible successor interpretations I_2 are, of course, the same as in Example 2. Rule r_1 could be changed to

$$r_1 \;=\; p(X) : 0.2 \vee q(X) : 0.8 \longleftarrow q(T, X)$$

to make it applicable whenever $\{q(X)\}$ succeeds in any earlier interpretation (not necessarily the previous one).

As the first-order Markov variant of CPT-L discussed in Sect. 2.2 is a special case of the more general variant discussed in this section, we will for the rest of the paper only consider full-history models. The conditional successor distribution $P(I_{t+1} \mid I_t, \ldots, I_0)$ will also be denoted by $P(I_{t+1} \mid I_{[0,t]})$.

3 Inference and parameter estimation in CPT-L

As for other probabilistic models, we can now formulate several computational tasks for the introduced CPT-L model:

- **Sampling**: sample sequences of interpretations $I_1, \ldots, I_T$ from a given CPT-theory $\mathcal{T}$ and initial interpretation I_0.
- **Inference**: given a CPT-theory $\mathcal{T}$ and a sequence of interpretations $I_0, \ldots, I_T$, compute $P(I_0, \ldots, I_T \mid \mathcal{T})$.
- **Parameter estimation**: given the structure of a CPT-theory $\mathcal{T}$ and a set D of sequences of interpretations, compute the maximum-likelihood parameters $\pi^* = \arg\max_\pi P(D \mid \pi)$, where π are the parameters of $\mathcal{T}$.
- **Prediction**: Let $\mathcal{T}$ be a CPT-theory, $I_0, \ldots, I_t$ a sequence of interpretations, and F a first-order query that represents a certain property of interest. Compute the probability that F holds at time $t+d$, that is, $P(I_{t+d} \models F \mid \mathcal{T}, I_0, \ldots, I_t)$.

Algorithmic solutions for solving these tasks will be presented in turn.

3.1 Sampling

Sampling from a CPT-theory is straightforward due to the causal semantics employed in the underlying CP-logic framework. Let $\mathcal{T}$ be a CPT-theory, and let I_0 be an initial interpretation. According to (4), we can sample from the joint distribution $P(I_1, \ldots, I_T \mid I_0)$ by successively sampling I_{t+1} from the distribution $P(I_{t+1} \mid I_{[0,t]})$ for $t = 0, \ldots, T-1$. This can be done directly using the constructive process that defines the semantics of CPT-L. We start with the empty interpretation $I_{t+1} = \{\}$, and first find all groundings $r\theta$ of rules $r \in \mathcal{T}$ that are applicable in $I_{[0,t]}$. For each grounded rule $r\theta$, we then randomly select one of its head elements $c \in \mathit{head}(r\theta)$ according to the probability distribution over head elements for that rule. The head element c is a conjunction of atoms, which need to be added to I_{t+1}. After adding all such conjuncts for all applicable rules, we have randomly sampled I_{t+1} from the desired distribution.

3.2 Inference for CPT-theories

Let $\mathcal{T}$ be a given CPT-theory, and $I_0, \ldots, I_T$ a sequence of interpretations. According to (4), the crucial task for solving the inference problem is to compute $P(I_{t+1} \mid I_{[0,t]})$ for $t = 0, \ldots, T-1$. According to (2), this involves marginalizing over all selections yielding I_{t+1} from $I_{[0,t]}$. However, the number of possible selections σ can be exponential in the number of ground rules $|\mathbf{R}_t|$ applicable in $I_{[0,t]}$, so a naive generate-and-test approach is infeasible. Instead, we present an efficient approach for computing $P(I_{t+1} \mid I_{[0,t]})$ without explicitly enumerating all selections yielding I_{t+1}, which is strongly related to the inference technique discussed in De Raedt et al. (2007). The problem of computing $P(I_{t+1} \mid I_{[0,t]})$ is first converted to a CNF formula over Boolean variables such that satisfying assignments correspond to selections yielding I_{t+1}. The formula is then compactly represented as a binary decision diagram (BDD), and $P(I_{t+1} \mid I_{[0,t]})$ efficiently computed from the BDD using dynamic programming. Although finding satisfying assignments for CNF formulae is a hard problem in general, the key advantage of this approach is that existing, highly optimized BDD software packages can be used.

The conversion of an inference problem $P(I_{t+1} \mid I_{[0,t]})$ to a CNF formula f is realized as follows:

1. Initialize $f := \mathit{true}$.
2. Let $\mathbf{R}_t$ denote the set of applicable ground rules in $I_{[0,t]}$. Rules $r \in \mathbf{R}_t$ are of the form $r = c_1 : p_1, \ldots, c_n : p_n \longleftarrow b_1, \ldots, b_m$, where c_i are conjunctions of literals (see Definition 2).

3. For all rules $r = c_1 : p_1, \ldots, c_n : p_n \longleftarrow b_1, \ldots, b_m$ in $\mathbf{R}_t$ do:
   (a) $f := f \wedge (r.c_1 \vee \cdots \vee r.c_n)$, where $r.c_i$ denotes a new (propositional) Boolean variable whose unique name is the concatenation of the name of the rule r with the head element c_i.
   (b) $f := f \wedge (\neg r.c_i \vee \neg r.c_j)$ for all $i \neq j$.
4. For all facts $l \in I_{t+1}$
   (a) Initialize $g := \mathit{false}$
   (b) for all $r \in \mathbf{R}_t$ and $c_i : p_i \in \mathit{head}(r)$ such that l is one of the atoms in the conjunction c_i do $g := g \vee r.c_i$
   (c) $f := f \wedge g$.
5. For all variables $r.c$ appearing in f such that one of the atoms in the conjunction c is not true in I_{t+1} do $f = f \wedge \neg r.c$.

A Boolean variable $r.c$ in f represents that head element c was selected in rule r. A selection σ thus corresponds to an assignment of truth values to the variables $r.c$, in which exactly one $r.c$ is true for every rule r. The construction of f ensures that all satisfying assignments for the formula f correspond to selections yielding I_{t+1}, and vice versa. Specifically, Step 3 of the algorithm assures that selections are obtained (that is, exactly one head element is selected per rule), Step 4 assures that the selection generates the interpretation I_{t+1}, and Step 5 assures that no facts are generated that do not appear in I_{t+1}. Thus, we have a one-to-one correspondence between satisfying assignments for the formula f and selections yielding I_{t+1}.

Example 4 The following formula f is obtained for the CPT-T theory given in Example 2 and the transition $\{p(a)\} \rightarrow \{p(a)\}$:

$$\underbrace{(r2.c_{21} \vee r2.c_{22}) \wedge (r3.c_{31} \vee r3.c_{32})}_{3.a} \wedge \underbrace{(\neg r2.c_{21} \vee \neg r2.c_{22})) \wedge (\neg r3.c_{31} \vee \neg r3.c_{32})}_{3.b}$$
$$\wedge \underbrace{(r2.c_{21} \vee r3.c_{31})}_{4} \wedge \underbrace{\neg r2.c_{22}}_{5}$$

where $c_{21} = p(a)$, $c_{22} = q(b) \wedge q(c)$, $c_{31} = p(a)$ and $c_{32} = \mathit{nil}$ are the head elements of rules r_2 and r_3. The parts of the formula are annotated with the steps in the construction algorithm that generated them.

From the formula f, a *reduced ordered binary decision diagram* (BDD) (Bryant 1986) is constructed. Let $x_1, \ldots, x_n$ denote an ordered set of Boolean variables (such as the $r.c$ contained in f). A BDD is a rooted, directed acyclic graph, in which nodes are annotated with variables and have out-degree 2, indicating that the variable is either true or false. Furthermore, there are two terminal nodes labeled with 0 and 1. Variables along any path from the root to one of the two terminals are ordered according to the given variable ordering. The graph compactly represents a Boolean function f over variables $x_1, \ldots, x_n$: given an instantiation of the x_i, we follow a path from the root to either 1 or 0 (indicating that f is true or false). Furthermore, the graph must be reduced, that is, it must not be possible to merge or remove nodes without altering the represented function. More formally, a BDD graph is said to be reduced if no further reduction operations can be applied. Reduction operations are (1) to merge any two isomorphic subgraphs in the BDD structure, and (2) to remove any node whose two children are isomorphic. It can be shown that reduced ordered BDD structures are a unique representation for any Boolean function, given a fixed variable ordering

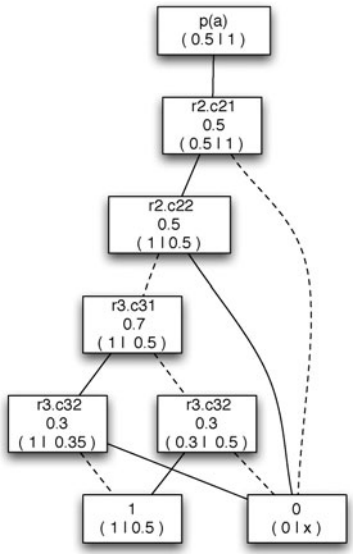

Fig. 2 BDD representing the formula f given in Example 4. The root node indicates the observed interpretation. The terminal nodes represent whether the path starting at the root node yields this interpretation. The other nodes are annotated with the rule r and head element c they represent, in the form of the Boolean variable $r.c$ used in f. If a node is left using a solid edge, the corresponding variable is assigned the value true, otherwise it is assigned the value false. Also given are upward probabilities $\alpha(N)$ and downward probabilities $\beta(N)$ for all nodes N, as $(\alpha(N) \mid \beta(N))$

(cf. Bryant 1986 for more details). Figure 2 shows the BDD resulting from the formula f given in Example 4.

From the BDD graph, $P(I_{t+1} \mid I_{[0,t]})$ can be computed in linear time using dynamic programming. The resulting algorithm is strongly related to the algorithm for inference in ProbLog theories (De Raedt et al. 2007), and will now be described in detail. First note that there is a one-to-one correspondence between paths in the BDD from the root to the 1-terminal and selections yielding I_{t+1}, where the path indicates which of the Boolean variables $r.c$ in f are assigned the value true, or equivalently, which head element c has been selected for rule r. To see this, consider Step 3 of the algorithm for converting a given inference problem into the BDD. It ensures that exactly one head element is chosen for every rule. Thus, in the BDD representation, every path to the 1-terminal must pass through all Boolean variables; otherwise, the state of one variable could be altered, violating the constraint encoded in Step 3 of the conversion algorithm.

We now recursively define for every node N in the BDD an *upward probability* $\alpha(N)$ as follows:

1. The upward probabilities for terminal nodes are defined as

$$\alpha(\text{0-terminal}) = 0 \quad \text{and} \quad \alpha(\text{1-terminal}) = 1.$$

2. Let N be a node in the BDD representing the Boolean variable $r.c$, with r a rule and c one of its head elements. Let N^-, N^1 denote the children of N, with N^- on the negative and N^+ on the positive branch. Then

$$\alpha(N) = \alpha(N^-) + P(c \mid r)\alpha(N^+).$$

Furthermore, we recursively define a *downward probability* $\beta(N)$ as follows:

1. The downward probability of the root node is defined as

$$\beta(root) = 1. \tag{5}$$

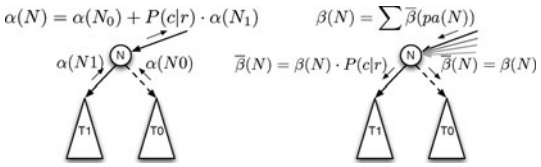

Fig. 3 Calculation of upward and downward probabilities for internal nodes in the BDD

2. Let N be a non-root node in the BDD. Let $N_1, \ldots, N_k$ denote the parents of N, with $N_1, \ldots, N_l = pa^+(N)$ reaching N by their positive branch and $N_{l+1}, \ldots, N_k = pa^-(N)$ reaching N by their negative branch. Then

$$\beta(N) = \sum_{i=1}^{l} \beta(N_i) P(c_i \mid r_i) + \sum_{i=l+1}^{k} \beta(N_i) \tag{6}$$

where $r_i.c_i$ is the Boolean variable associated with node N_i.

The definition of upward and downward probabilities is visualized in Fig. 3. The values $\alpha(N)$ and $\beta(N)$ can be interpreted as probabilities of partial selections, which are determined by the path from the 1-terminal (α) or the root (β) to the node N. They roughly correspond to the forward-backward probabilities used for inference in hidden Markov models (Rabiner 1989), or inside-outside probabilities used in stochastic context free grammars.

The following theorem states that the desired probability $P(I_{t+1} \mid I_{[0,t]})$ for inference can be easily obtained given the upward and downward probabilities:

Theorem 2 *Let $\mathcal{B}$ be a BDD resulting from the conversion of an inference problem $P(I_{t+1} \mid I_{[0,t]})$, annotated with upward and downward probabilities as defined above, and let*

$$\Gamma = \{\sigma \mid I_{[0,t]} \xrightarrow{\sigma} I_{t+1}\}$$

be the set of selections yielding I_{t+1}. Then

$$\begin{aligned} \alpha(root) &= \sum_{\sigma \in \Gamma} P(\sigma) \\ &= P(I_{t+1} \mid I_{[0,t]}). \end{aligned} \tag{7}$$

A proof of the theorem is given in Appendix A. Note that the downward probabilities will only be needed for the parameter estimation algorithm discussed in Sect. 3.4. Computing upward and downward probabilities from their recursive definitions is straightforward, thus Theorem 2 concludes the description of the BDD-based inference algorithm for CPT-L.

Computational costs are linear in the size of the BDD graph. The efficiency of this method thus crucially depends on the size of this graph, which in turn depends strongly on the chosen variable ordering $x_1, \ldots, x_n$. Unfortunately, computing an optimal variable ordering is NP-hard. However, existing implementations of BDD packages[1] contain sophisticated heuristics to find a good ordering for a given function in polynomial time.

[1] Our implementation uses the CUDD package http://vlsi.colorado.edu/~fabio/CUDD/.

3.3 Partially lifted inference for CPT-theories

We have so far specified CPT-L theories using first-order logic, but carried out inference at the ground level. This is a common strategy in many statistical relational learning frameworks: the first-order model specification serves as a template language from which a ground model is constructed for inference. A popular approach is to use graphical models as ground models. These can be directed (as in Relational Bayesian Networks (Jaeger 1997), Bayesian Logic Programs (Kersting and De Raedt 2007), or CP-logic (Vennekens et al. 2006)), or undirected (as in Markov Logic Networks (Richardson and Domingos 2006)).

In CPT-L, the grounded inference problem takes the form of a (propositional) Boolean formula, for which we need to compute all satisfying assignments. This problem can be solved efficiently using binary decision diagrams, as shown in Sect. 3.2. However, the size of the inference problem (and resulting BDD) depends on the size of the grounded first-order model, which can be large compared to the original first-order model specification. Recent work on *lifted inference* in first-order models (see, for example, Poole (2003) and Milch et al. (2008)) has shown that computational efficiency can be improved significantly if inference is performed directly at the first-order level. We now discuss a lifted inference algorithm for CPT-theories. The general idea is to solve a part of the overall inference problem directly at the first-order level, without compiling it into the binary decision diagram. The approach is best illustrated using an example:

Example 5 Reconsider the CPT-Theory given in Example 2. Suppose we want to compute the probability $P(I_{t+1} \mid I_{[0,t]})$, where $I_t = \{q(a), q(b), p(1), p(2), p(3)\}$, $I_{t+1} = \{p(a), p(b)\}$, and $I_{[0,t-1]}$ are irrelevant as the theory refers only to the previous time-point. Rules r_1 and r_3 are applicable, and

$$\mathbf{R}_t = \{r_1\{X/a\}, r_1\{X/b\}, r_3\{X/1\}, r_3\{X/2\}, r_3\{X/3\}\}.$$

We need to compute

$$P(I_{t+1} \mid I_t) = \sum_{\sigma \in \Gamma} P(\sigma) \tag{8}$$

where Γ is the set of selections yielding I_{t+1} from I_t. Computing this sum over probabilities of selections $\sigma \in \Gamma$ is the ground inference problem, which can be solved using BDDs as explained in Sect. 3.2. According to (1), the $P(\sigma)$ are of the form

$$P(\sigma) = f_{11} f_{12} f_{31} f_{32} f_{33}$$

where $f_{11}, f_{12} \in \{0.2, 0.8\}$ are the probabilities of selected head elements of ground rules $r_1\{X/a\}, r_1\{X/b\} \in \mathbf{R}_t$, and $f_{31}, f_{32}, f_{33} \in \{0.7, 0.3\}$ are the probabilities of selected head elements of ground rules $r_3\{X/a\}, r_3\{X/b\}, r_3\{X/c\} \in \mathbf{R}_t$.

However, inspecting rule r_1 and I_{t+1}, we see that irrespective of the substitution θ grounding rule r_1 in I_t, only the first head element of r_1 can be used in a selection. Thus, factors f_{11} and f_{12} are always 0.2, and (8) simplifies to

$$P(I_{t+1} \mid I_t) = 0.2 \cdot 0.2 \sum_{\sigma' \in \Gamma'} P(\sigma') \tag{9}$$

where σ' only selects head elements for rule r_3. That is, $P(\sigma')$ is of the form

$$P(\sigma') = f_{31} f_{32} f_{33}.$$

Note that the remaining ground inference problem—summing over the partial selections σ'—is smaller than the original one given by (8). The remaining problem can be solved using the BDD-based inference method as explained above. However, when converting this inference problem to a Boolean formula f, we need to take into account that some facts appearing in the next interpretation I_{t+1} have already been generated by the head elements selected for groundings of rule r_1, and thus do not need to be generated anymore by groundings of rule r_3. That is, we simply ignore already generated facts in Step 4 of the construction of f.

In fact, we can go one step further, and note that also for rule r_3 we can determine the selected head element irrespective of the substitution used to ground the rule in I_t. It is easily determined by logical inference that head element $p(X)$ cannot be grounded in I_{t+1} given that the body $p(X)$ is grounded in I_t, thus only the second head element can be used for any grounding of rule r_3 in any selection σ'. Thus, (9) is further simplified to

$$\begin{aligned} P(I_{t+1} \mid I_t) &= 0.2 \cdot 0.2 \cdot 0.3 \cdot 0.3 \cdot 0.3 \\ &= 0.2^{K_{r_1}} 0.3^{K_{r_3}} \end{aligned}$$

where K_{r_i} is the number of groundings of rule r_i in I_t.

The key observation in the above example is that for both r_1 and r_3 we could logically infer the head element used in any selection $\sigma \in \Gamma$ under any grounding of the rules in I_t. Note that in general, only a subset of the rules can be removed from the ground inference problem in this way.

Generalizing from Example 5, we can describe the partially lifted inference algorithm for any given CPT-theory $\mathcal{T} = \{r_1, \ldots, r_k\}$ and inference problem $P(I_{t+1} \mid I_{[0,t]})$ as follows:

1. Let $\mathbf{R}_t$ denote the set of all ground rules applicable in $I_{[0,t]}$.
2. Define

$$\overline{\mathbf{R}}_t = \{r\theta \in \mathbf{R}_t \mid I_{t+1},\ I_{[0,t]} \text{ logically determine the head element selected for } r\theta\}$$

   For a rule $r\theta \in \overline{\mathbf{R}}_t$, let $\overline{\sigma}(r\theta)$ denote the head element that must be selected.
3. Compute

$$\begin{aligned} P(I_{t+1} \mid I_{[0,t]}) &= \prod_{r\theta \in \overline{\mathbf{R}}_t} P(\overline{\sigma}(r\theta) \mid r\theta) \sum_{\sigma' \in \Gamma'} P(\sigma') \\ &= \prod_{r \in \mathcal{T}} \prod_{c_r \in head(r)} P(c_r \mid r)^{K_{r,c}} \sum_{\sigma' \in \Gamma'} P(\sigma'), \end{aligned} \tag{10}$$

   where

$$K_{r,c} = |\{r\theta \in \overline{\mathbf{R}}_t \mid \overline{\sigma}(r\theta) = c\theta\}|$$

   and Γ' is the set of selections of head elements for rules in $\mathbf{R}_t \setminus \overline{\mathbf{R}}_t$ that yield I_{t+1} from $I_{[0,t]}$, given that we select head element $\overline{\sigma}(r\theta)$ for rules in $r\theta \in \overline{\mathbf{R}}_t$. Note that in (10) we have integrated all ground rules for which a particular head element c_r has to be selected into one factor, which has to be taken to the power of $K_{r,c}$, namely the number of such ground rules. Thus, we have performed a partially lifted probability calculation.

The set $\overline{\mathbf{R}}_t$ contains those grounded rules $r\theta$ for which we can prove—using logical inference on $body(r\theta)$, $head(r\theta)$, and the interpretations $I_{[0,t]}$ and I_{t+1}—that a particular head element $\overline{\sigma}(r\theta)$ has to be selected for $r\theta$. For instance, all groundings of rule r_1 in Example 5 are in this set, because no ground facts of the form $q(X)\theta$ appear in I_{t+1}, and thus the first head element of r_1 always has to be selected. In fact, for Example 5 we have $\overline{\mathbf{R}}_t = \mathbf{R}_t$. The term $K_{r,c}$ is the number of groundings of a rule $r \in \mathcal{T}$ for which we know that the head element $c \in head(r)$ is selected for the grounded rule. For instance, in Example 5, $K_{r_1,p(X)} = 2$ and $K_{r_3,nil} = 3$. In practice, the counting variables $K_{r,c}$ can be computed as follows. For each rule r, we first determine the set of groundings θ such that $r\theta$ holds in $I_{[0,t]}$ and exactly one of the grounded head elements holds in I_{t+1}; this can be achieved with a single logical query. We then count for each head element $c_r \in head(r)$ the number of times the unique grounded head determined in the first step was subsumed by c_r, this yields the term $K_{r,c}$.

Comparing the outlined partially lifted inference algorithm to other lifted inference algorithms proposed in the literature, such as first-order probabilistic inference (Poole 2003) or lifted inference with counting formulas (Milch et al. 2008), we note that it is much simpler and, correspondingly, more limited in scope. Nevertheless, it proved surprisingly effective in our experimental evaluation (see Sect. 4).

Note that the efficiency of the presented inference algorithm depends on the fact that the selection of a particular head element is enforced by a given successor interpretation. This in turn depends on the closed-world assumption, which states that any atom not observed is false.

3.4 Parameter estimation

Assume the structure of a CPT-theory is given, that is, a set $\mathcal{T} = \{r_1, \ldots, r_k\}$ of rules of the form

$$r_i = (c_{i1} : p_{i1}) \vee \cdots \vee (c_{in_i} : p_{in_i}) \longleftarrow b_{i1}, \ldots, b_{im_i},$$

where $\pi = \{p_{ij}\}_{i,j}$ are the unknown parameters to be estimated from a set of training sequences $\mathcal{D}$. A standard approach is to find maximum-likelihood parameters

$$\pi^* = \arg\max_{\pi} P(\mathcal{D} \mid \pi),$$

that is, to set the parameters such that we maximize the probability of generating the data $\mathcal{D}$ from $\mathbf{T}$. When generating $\mathcal{D}$ from $\mathcal{T}$, a rule $r_i \in \mathbf{T}$ is typically applied multiple times: in the form of different groundings $r_i\theta$, and in different transitions (appearing in different training sequences). We would like to set

$$\forall i, j : p_{ij} = \frac{\kappa_{ij}}{\sum_{l=1}^{n_i} \kappa_{il}}, \tag{11}$$

where κ_{ij} denotes the number of times head element c_{ij} was selected in any application of the rule r_i while generating $\mathcal{D}$. However, the quantity κ_{ij} is not directly observable. To see why this is so, first consider a single transition $I_{[0,t]} \rightarrow I_{t+1}$ in one training sequence. We know the set of rules $\mathbf{R}_t$ applied in the transition; however, there are in general many possible selections σ of rule head elements yielding I_{t+1}. The information about which selection was used, that is, which rule has generated which fact in I_{t+1}, is hidden. We will now derive an efficient Expectation-Maximization algorithm in which the unobserved variables are the selections used at a transition, and κ_{ij} the sufficient statistics. To this aim, we first need to

compute expected values of the κ_{ij} given the observations and the current model parameters π, and then re-estimate π according to (11) where the κ_{ij} are replaced by their expectation.

To keep the notation uncluttered, we first consider a single transition $\Delta = I_{[0,t]} \to I_{t+1}$. Let $\mathbf{R}_t$ denote the set of rules applicable in the transition, and let $\kappa_{ij}^{\theta} \in \{0, 1\}$ denote whether the grounded head element $c_{ij}\theta$ was selected in the application of a grounded rule $r_i\theta \in \mathbf{R}_t$. Let furthermore $\Gamma = \{\sigma \mid I_{[0,t]} \xrightarrow{\sigma} I_{t+1}\}$ be the set of selections yielding I_{t+1}. For a given selection $\sigma \in \Gamma$, we have

$$\kappa_{ij}^{\theta} = \begin{cases} 1, & \sigma(r_i\theta) = c_{ij}\theta \\ 0, & \text{otherwise,} \end{cases} \tag{12}$$

and

$$\kappa_{ij} = \sum_{\theta : r_i\theta \in \mathbf{R}_t} \kappa_{ij}^{\theta} \tag{13}$$

where the sum runs over all groundings $r_i\theta \in \mathbf{R}_t$ of rule r_i. However, the selection σ is not observed, thus we instead have to consider the expectation $\mathbb{E}[\kappa_{ij} \mid \pi, \Delta]$ of κ_{ij} with respect to the posterior distribution $P(\sigma \mid \pi, \Delta)$ over selections given the data and current parameters. It holds that

$$\begin{aligned} \mathbb{E}[\kappa_{ij}^{\theta} \mid \pi, \Delta] &= P(\kappa_{ij}^{\theta} = 1 \mid \pi, \Delta) \\ &= \sum_{\sigma \in \Gamma} P(\kappa_{ij}^{\theta} = 1 \mid \sigma) P(\sigma \mid \pi, \Delta) \end{aligned} \tag{14}$$

where $P(\kappa_{ij}^{\theta} = 1 \mid \sigma) \in \{0, 1\}$ according to (12). Equation (13) now implies

$$\mathbb{E}[\kappa_{ij} \mid \pi, \Delta] = \sum_{\theta : r_i\theta \in \mathbf{R}_t} \mathbb{E}[\kappa_{ij}^{\theta} \mid \pi, \Delta],$$

which concludes the expectation step of the Expectation-Maximization algorithm for a single transition Δ. If the data $\mathcal{D}$ contains multiple transitions (possibly appearing in multiple sequences), we can simply sum up the quantities $\mathbb{E}[\kappa_{ij} \mid \pi, \Delta]$ for each transition. Finally, given the expectation of the sufficient statistics κ_{ij}, the maximization step in EM is

$$p_{ij}^{(new)} = \frac{\mathbb{E}[\kappa_{ij} \mid \pi, \mathcal{D}]}{\sum_j \mathbb{E}[\kappa_{ij} \mid \pi, \mathcal{D}]}.$$

As usual, expectation and maximization steps are iterated until convergence in likelihood space.

The key algorithmic challenge in the outlined EM algorithm is to compute the expectation given by (14) efficiently. Note that this again involves summing over all selections yielding the next interpretation, much as in the inference problem discussed in Sects. 3.2 and 3.3. In fact, the quantity $\mathbb{E}[\kappa_{ij}^{\theta} \mid \pi, \Delta]$ can also be obtained from the upward and downward probabilities introduced in Sect. 3.2. More formally, the following holds:

Theorem 3 *Let p_{ij} be the parameter associated with head element c_{ij} in rule r_i, let $\Delta = I_{[0,t]} \to I_{t+1}$ be a single transition, and let $r_i\theta \in \mathbf{R}_t$ denote a grounding of r_i applicable in $I_{[0,t]}$. Let $N_1, \ldots, N_k$ be all nodes in the BDD associated with the Boolean variable $r_i\theta.c_{ij}\theta$*

resulting from the grounded rule $r_i\theta$, and let N_l^+ be the child on the positive branch of N_l. Then

$$\mathbb{E}[\kappa_{ij}^{\theta} \mid \pi, \Delta] = \frac{1}{P(I_{t+1} \mid I_{[0,t]})} \sum_{l=1}^{k} \beta(N_l) p_{ij} \alpha(N_l^+). \tag{15}$$

As for the inference problem discussed in Sect. 3.2, we can thus compute the estimation step given by (14) in time linear in the size of the BDD. The theorem can be proven using similar techniques as in the proof of Theorem 2; however, the proof is slightly more involved and thus moved to Appendix A.

Finally, note that at this point we can again make use of the partial lifted inference algorithm discussed in Sect. 3.3. A part of the expectation computation is then solved directly at the first-order level, while the rest is solved using dynamic programming in the BDD as explained above.

Note that the presented algorithms for inference and parameter estimation can be significantly more efficient than the corresponding algorithms in the more general CP-logic framework. Specifically, in CP-logic the inference and learning problems basically have to be grounded into a Bayesian Network, which can grow very large depending on the characteristics of the domain. This often makes (exact) inference computationally challenging. In contrast, the inference and learning techniques we discussed here take advantage of the particular problem setting and model structure (that is, sequential and fully observable data). The experimental evaluation presented in Sect. 4 indeed shows that with these techniques we can perform exact inference in only seconds, for problems where the ground Bayesian Network would contain hundreds of thousands of nodes.

3.5 Prediction

Assume we are given an observation sequence $I_0, \ldots, I_t$, a CPT-theory $\mathcal{T}$, and a property of interest F (represented as a first-order query), and would like to compute $P(I_{t+d} \models F \mid I_0, \ldots, I_t, \mathcal{T})$. For instance, a robot might like to know the probability that a certain world state is reached at time $t + d$, given its current world model and observation history. Or, in the MMOG domain, we might want to compute the probability that a particular player will have won the game at time $t + d$, given a model of game dynamics and an observation history. We will assume that F is any first-order query that could be posed to a logic programming system such as Prolog, making use of the available background knowledge B.

Powerful statistical relational learning systems are in principle able to compute the quantity $P(I_{t+d} \models F \mid I_0, \ldots, I_t, \mathcal{T})$ exactly by "unrolling" the world model into a large dynamic graphical model. However, this is computationally expensive as it requires to marginalize out all (unobserved) intermediate world states $I_{t+1}, \ldots, I_{t+d-1}$, and thus often not practical in complex worlds. In contrast, inference in CPT-theories draws its efficiency from the full observability assumption, as outlined in Sect. 3. As an alternative to the "unrolling" approach, we thus propose a straightforward sample-based approximation to compute $P(I_{t+d} \models F \mid I_t, \mathcal{T})$ that preserves the efficiency of our approach. The idea is to obtain independent samples from the Boolean random variable $I_{t+d} \models F$ given $\mathcal{T}$ and $I_0, \ldots, I_t$, and estimate the desired probability as the fraction of positive samples.

Given $I_0, \ldots, I_t$, it is straightforward to obtain independent samples of the conditional distribution $P(I_{t+1}, \ldots, I_{t+d} \mid I_0, \ldots, I_t, \mathcal{T})$ by forward sampling from the stochastic process represented by $\mathcal{T}$, as explained in Sect. 3.1. Ignoring $I_{t+1}, \ldots, I_{t+d-1}$, we can simply check whether $I_{t+d} \models F$ in the sampled interpretation I_{t+d}. After repeatedly sampling

interpretations $I^{(1)}_{t+d}, \ldots, I^{(K)}_{t+d}$ in this fashion, the fraction of $I^{(k)}_{t+d}$ for which $I^{(k)}_{t+d} \models F$ is then an unbiased estimator of the true probability $P(I_{t+d} \models F \mid I_t, \mathcal{T})$, and will in fact quickly converge towards this true probability for large K.

4 Experimental evaluation[2]

In this section, we experimentally validate the proposed CPT-L approach in several (artificial and real-world) domains as well as in different learning settings. The general setting discussed in this paper, namely constructing models for stochastic processes with complex state representations, covers a wide range of application domains. It is appropriate whenever systems evolve over time and are complex enough that their states cannot easily be described using a propositional representation. A prominent example are states that are characterized by a graph structure relating different agents and/or world artifacts at a given point in time (as in dynamic social networks, computer networks, the world wide web, games, marketplaces, et cetera). In this setting, observations consist of sequences of labeled (hyper)graphs, cf. Fig. 8. To experimentally evaluate CPT-L, we have selected the following domains as representative examples:

Stochastic blocks world domain This domain is a stochastic version of the well-known artificial *blocks world* domain, representing an agent that is moving blocks which are stacked on a table. We use this artificial domain to perform controlled experiments, testing the scaling and convergence behavior of inference and learning algorithms.

Chat room domain This domain is concerned with the analysis of user interaction in chat rooms. We have monitored a number of IRC chat rooms in real time, and recorded who was sending messages to whom using the PieSpy utility (Mutton 2004). This results in dynamically changing graphs of user interaction, representing the social network structure among chat room participants, cf. Fig. 5. We learn these dynamics using separate models for different chat rooms. The resulting set of models can be used to visualize commonalities and differences in the behavior displayed in different chat rooms, thereby characterizing the underlying user communities.

Massively multiplayer online game domain As a final evaluation domain introduced in Thon et al. (2008), we consider the large-scale massively multiplayer online strategy game Travian.[3] Game worlds feature thousands of players, game artifacts such as cities, armies, and resources, and social player interaction in alliances. Game states in Travian are complex and richly structured, and transitions between game states highly stochastic as they are determined by player actions. We have logged the state of a "live" game server over several months, recording high-level game states as visualized in Fig. 8. We address different learning tasks in the Travian domain, such as predicting player actions (prediction setting) and identifying groups of cooperating alliances (classification setting).

The goal of our experimental study is two-fold. First, we want to evaluate the effectiveness of the proposed approach. That is, we explore whether it is possible to learn dynamic stochastic models for the above-mentioned relational domains, and to solve the resulting

[2]The implementation, models and data will be made available at http://www.ingothon.de/.

[3]www.travian.com; www.traviangames.com.

inference, prediction, and classification tasks. Our second goal is to evaluate the efficiency of the proposed algorithms. That is, we will evaluate the scaling behavior for domains with a large number of objects and relationships, and in particular explore the advantage of performing partially lifted inference in such domains. Experiments to address these questions will be presented in turn for the three outlined evaluation domains in the rest of this section.

4.1 Experiments in the stochastic blocks world domain

As an artificial testbed for CPT-L, we performed experiments in a stochastic version of the well-known *blocks world* domain. The domain was chosen because it is truly relational and also serves as a popular artificial world model in agent-based approaches such as planning and reinforcement learning. Moreover, application scenarios involving agents that act and learn in an environment are one of the main motivations for CPT-L.

World model The blocks world we consider consists of a table and a number of blocks. Every block rests on exactly one other block or the table, denoted by a fact $on(A, B)$. Blocks come in different sizes, denoted by $size\_of(A, N)$ with $N \in \{1, \ldots, 4\}$. A predicate $free(B) \longleftarrow not(on(A, B))$ is defined in the background knowledge. Additionally, a background predicate $stack(A, S)$ defines that block A is part of a stack of blocks, which is represented by its lowest block S. Actions in the blocks world domain are of the form $move(A, B)$. If both A and B are free, the action moves block A on B with probability $1 - \epsilon$, with probability ϵ the world state does not change. Furthermore, a stack S can start to jiggle, represented by $jiggle(S)$. A stack can start to jiggle if its top block is lifted, or a new block is added to it. Furthermore, stacks can start jiggling without interference from the agent, which is more likely if they contain many blocks and large blocks are stacked on top of smaller ones. Stacks that jiggle collapse in the next time step, and all their blocks fall on the table. Two example rules from this domain are

$$(jiggle(S) : 0.2) \vee (nil : 0.8) \longleftarrow move(A, B), stack(A, S)$$

$$(jiggle(S) : 0.2) \vee (nil : 0.8) \longleftarrow move(A, B), stack(B, S),$$

they describe that stacks can start to jiggle if blocks are added to or taken from a stack. Furthermore, we assume the agent follows a simple policy that tries to build a large stack of blocks by repeatedly stacking the free block with second-lowest ID on the free block with lowest ID. This strategy would result in one large stack of blocks if stacks never collapsed. In our experiments, the policy was supplied as background knowledge, that is, the predicate $move/2$ was hard-coded by a logical definition in the background knowledge and not part of the learning problem. The model had 14 rules with 24 parameters in total.

Results in the blocks-world domain In a first experiment, we explore the convergence behavior of the EM algorithm for CPT-L. The world model together with the policy for the agent, that specifies which block to stack next, is implemented by a (gold-standard) CPT-theory $\mathcal{T}$, and a training set of 20 sequences of length 50 each is sampled from $\mathcal{T}$. From this data, the parameters are re-learned using EM. Figure 4, left graph, shows the convergence behavior of the algorithm on the training data for different numbers of blocks in the domain, averaged over 15 runs. It shows rapid and reliable convergence. Figure 4, right graph, shows the running time of EM as a function of the number of blocks. The scaling behavior is roughly linear, indicating that the model scales well to reasonably large domains. Absolute running times are also low, with about 1 minute for an EM iteration in a world with 50

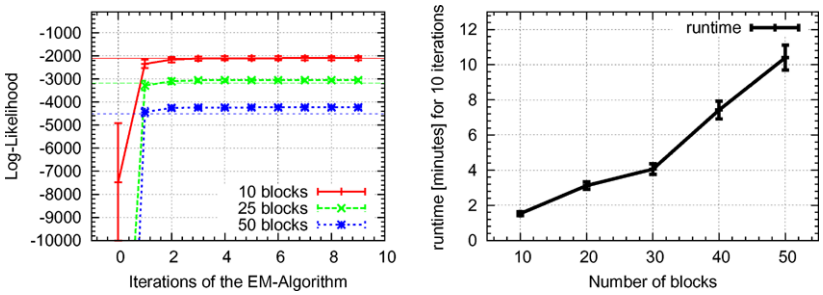

Fig. 4 *Left graph*: per-sequence log-likelihood on the training data as a function of the EM iteration. *Right graph*: Running time of EM as a function of the number of blocks in the world model

blocks.[4] This is in contrast to other, more expressive modeling techniques which typically scale badly to domains with many objects. The theory learned (Fig. 4) is very close to the ground truth ("gold standard model") from which training sequences were generated. On an independent test set (also sampled from the ground truth), log-likelihood for the gold standard model is −4510.7, for the learned model it is −4513.8, while for a theory with randomly initialized parameters it is −55999.4 (50 blocks setting). Manual inspection of the learned model also shows that parameter values are on average very close to those in the gold-standard model.

The experiments presented so far show that relational stochastic domains of substantial size can be represented in CPT-L. The presented algorithms are efficient and scale well in the size of the domain, and show robust convergence behavior.

4.2 Experiments in the chat room domain

For our experiments in the chat room domain, we have selected the following 7 well-frequented IRC chat rooms: *football@irc.efnet.net*, *iphone@irc.efnet.net*, *computer@irc.efnet.net*, *poker@irc.efnet.net*, *math@irc.efnet.net*, *politics@irc.efnet.net*, and *travian@irc.travian.org*. Each chat room was monitored for one day using the PieSpy utility (Mutton 2004), generating a sequence of user interaction graphs as those shown in Fig. 5. For each chat room, we selected the first 100 observations in the sequence of user interaction graphs as a single observation sequence for that chat room, yielding 7 observation sequences $S_1, \ldots, S_7$.

We have again hand-coded a simple CPT-theory $\mathcal{T}$ for this domain, which makes use of a number of graph-theoretic properties defined in the background knowledge, such as graph centrality, node degree, closeness, betweenness, and co-citation. As an example rule, consider

$$communicates(P1, P2) : 0.1 \vee nil : 0.9$$
$$\longleftarrow cocitation(P1, P2, CC), \neg communicates(P1, P2), \neg communicates(P2, P1)$$

encoding that two chat participants start talking to each other if there is a third participant with whom they have both talked before. The following three rules encode that a random

[4] All experiments were run on standard PC hardware, 2.4 GHz Intel Core 2 Duo processor, 1 GB memory.

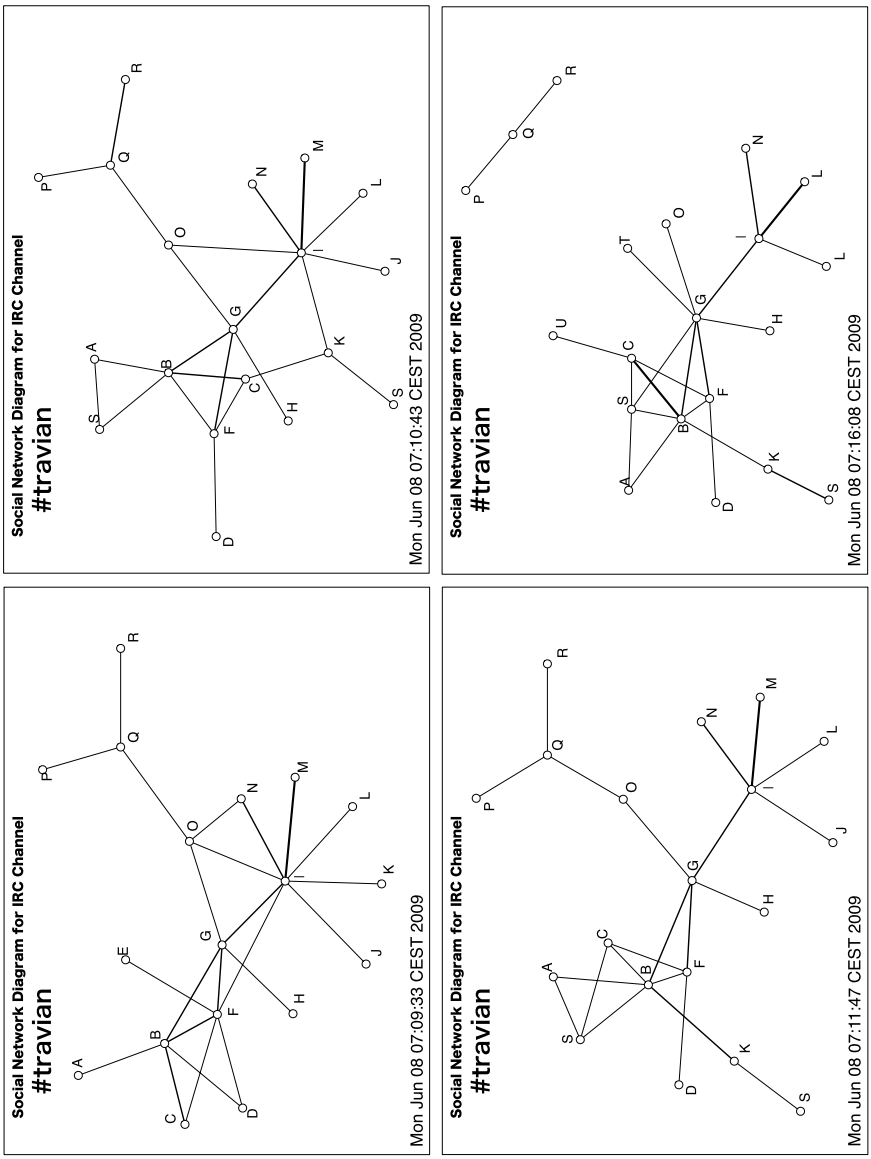

Fig. 5 User interaction graphs from the Chat Room Domain. Shown are four different time points during the observation sequence recorded for the irc.travian.org chat room

person starts to communicate with another person which has above average *betweeness*, *degree*, or *closeness*.

$$communicates(P1, P2) : 0.1 \vee nil : 0.9$$
$$\longleftarrow betweeness(P1, C1), avg\_betweeness(Avg), C1 > Avg,$$
$$\neg communicates(P1, P2), \neg communicates(P2, P1)).$$
$$communicates(P1, P2) : 0.1 \vee nil : 0.9$$
$$\longleftarrow degree(P1, C1), person(P2), avg\_degree(Avg), C1 > Avg,$$
$$\neg communicates(P1, P2), \neg communicates(P2, P1).$$
$$communicates(P1, P2) : 0.1 \vee nil : 0.9$$
$$\longleftarrow closeness(P1, C1), person(P2), avg\_closeness(Avg), person(P1), C1 > Avg,$$
$$\neg communicates(P1, P2), \neg communicates(P2, P1).$$

In the model definition rule heads also contain a third head element for reversed communication direction $communicates(P2, P1)$, which was omitted above for increased readability. In total the model had 7 rules with 11 parameters (note that a rule with three head elements has two parameters, as parameters must sum to one).

For each chat room we learn the parameters of the CPT-theory $\mathcal{T}$ using the EM algorithm presented in Sect. 3.4, resulting in 7 CPT-theories $\mathcal{T}_1, \ldots, \mathcal{T}_7$ with the same rule structure but different parameters. Learning took about 10 seconds per theory $\mathcal{T}_i$. The learned CPT-theories can be seen as a probabilistic representation of the typical interaction behavior among members of that chat room, reflecting the corresponding different user communities. For instance, they could represent how quickly the interaction graph changes, the degree of connectivity in the interaction graph, or how large the fluctuation in chat participants is over time. The goal of our experiment is to visualize the commonalities and differences in the behavior of these different user groups. To this end, we have evaluated the likelihood $P(S_i \mid \mathcal{T}_j)$ of each sequence S_i under the learned CPT-theory $\mathcal{T}_j$. This gives an indication as to how well the behavior in chat room i is explained by the model learned for chat room j, thus indicating the similarity in user behavior for the corresponding two communities.

The result of this experiment is visualized in Fig. 6. We can distinguish different clusters of chat rooms, or, equivalently, user communities. For instance, chat rooms that are concerned with recreational topics such as *travian@irc.travian.org* and *football@irc.efnet.net* (as well as *iphone@irc.efnet.net*) are clearly distinguishable from chat rooms concerned with more "serious" topics such as *math@irc.efnet.net* and *politics@irc.efnet.net*. Manual inspection of the learned rule parameters showed that in the "serious" chat domains the likelihood of a communication between two players mostly depends on the betweenness and degrees of the nodes involved, while in the "recreational" chats shared cocitations are more important.

4.3 Experiments in the massively multiplayer online game domain

We now report on experiments in *Travian* domain. In Travian, players are spread over several independent *game worlds*, with approximately 20.000–30.000 players interacting in a single world. Travian gameplay follows a classical strategy game setup. A game world consists of

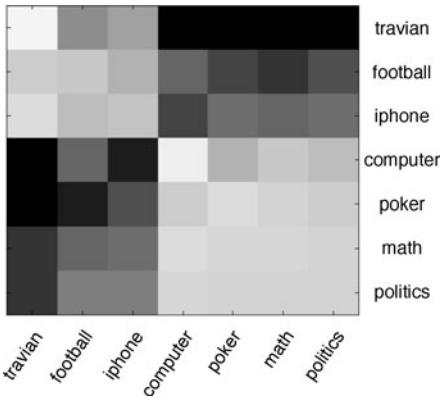

Fig. 6 Plot of the likelihood $P(S_i \mid \mathcal{T}_j)$ of a sequence S_i (corresponding to chat room i) under the CPT-theory $\mathcal{T}_j$ (learned on chat room j). *Rows* correspond to models $\mathcal{T}_j$ and *columns* to sequences S_i. *Lighter colors* indicate higher likelihoods

a large *grid-map*, and each player starts with a single *city* located on a particular tile of the map. During the course of the game, players harvest *resources* from the environment, improve their cities by construction of *buildings* or research of *technologies*, or found new cities on other (free) tiles of the map. Additionally, players can build different military units which can be used to attack and conquer other cities on the map, or trade resources on a global marketplace.

In addition to these low-level gameplay elements, there are high-level aspects of gameplay involving *multiple players*, which need to cooperate and coordinate their playing to achieve otherwise unattainable game goals. More specifically, in Travian players dynamically organize themselves into *alliances*, for the purpose of jointly attacking and defending, trading resources or giving advice to inexperienced players. Such alliances constitute social networks for the players involved, where diplomacy is used to settle conflicts of interests and players compete for an influential role in the alliance. In the following, we will take a high-level view of the game and focus on modeling player interaction and cooperation in alliances rather than low-level game elements such as resources, troops and buildings. Figure 7 shows such a high-level view of a (partial) Travian game world, represented as a graph structure relating cities, players and alliances which we will refer to as a *game graph*. It shows that players in one alliance are typically concentrated in one area of the map—traveling over the map takes time, and thus there is little interaction between players far away from each other.

We are interested in the *dynamic* aspect of this world: as players are acting in the game environment (e.g. by conquering other players' cities and joining or leaving alliances), the game graph will continuously change, and thereby reflect changes in the social network structure of the game. As an example for such transition dynamics, consider the sequence of game graphs shown in Fig. 8. Here, three players from the red alliance launch a concerted attack against territory currently held by the blue and yellow alliances, and partially conquer it.

Data collection and preprocessing The data used in the experiments was collected from a "live" Travian server with approximately 25.000 active players. Over a period of three months (December 2007, January 2008, February 2008), high-level data about the current state of the game world was collected once every 24 hours. This included information about all cities, players, and the alliance structure in the game. For cities, their size and position on the map are available; for players, the list of cities they own; and for alliances the list of players currently affiliated with that alliance.

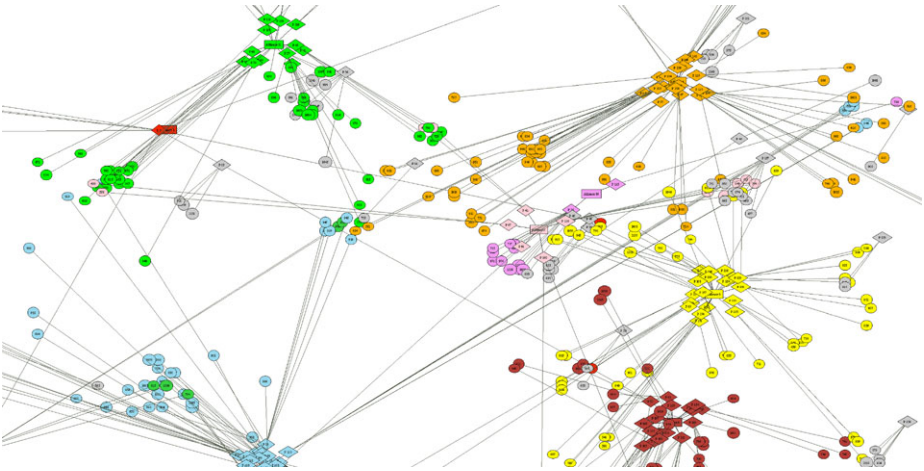

Fig. 7 High-level view of a (partial) game world in Travian. *Circular nodes* indicate cities, shown in their true positions on the game's grid-map. *Diamond-shaped nodes* indicate players, and are connected to all cities currently owned by the player. *Rectangular nodes* indicate alliances, and are connected to all players currently members of the alliance. (The alliance affiliation is additionally indicated by color-coding of the cities and players)

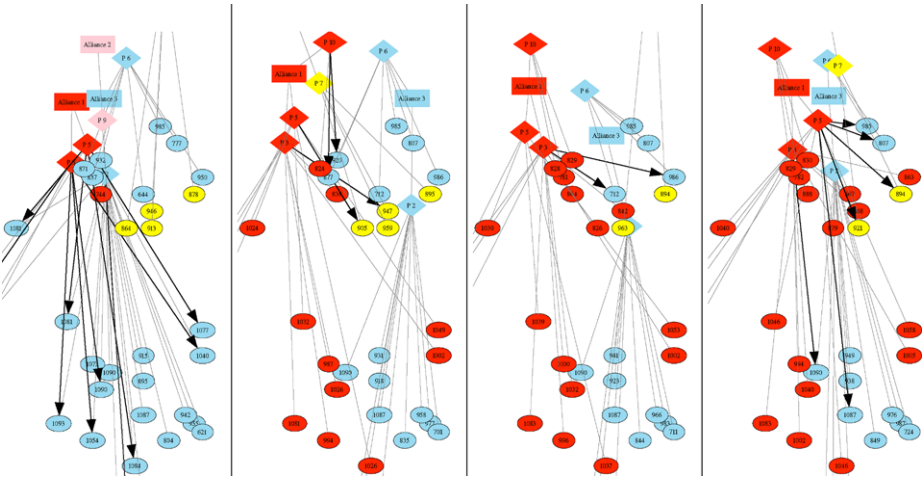

Fig. 8 Travian game dynamics visualized as changes in the game graph (for $t = 1, 2, 3, 4, 5$). *Bold arrows* indicate conquest attacks by a player on a particular city

The game data was represented using predicates $city(C, X, Y, S, P)$ (city C of size S at coordinates X, Y held by player P), $allied(P, A)$ (player P is a member of alliance A), $conq(P, C)$ (indicating a conquest attack of player P on city C) and $alliance\_change(P, A)$ (player P changes affiliation to alliance A). A predicate $distance(C_1, C_2, D)$ with $D \in \{near, medium, far\}$ computing the (discretized) distance between cities was defined in the background knowledge. Sequences consist of between 29 and 31 such state descriptions.

Classification experiments As a classification setting, we consider the problem of identifying so-called *meta-alliances* in Travian, which was recently introduced by Karwath et al. (2008). A meta-alliance is a group of alliances that closely cooperate, thereby allowing large groups of players to work together. We manually identified meta-alliances in the collected game data based on the alliance names (a small free-text field). For instance, it is easy to recognize that the alliances *'.~A~.'*, *'.=A=.'*, and *'.-A-.'* are different wings of the same meta-alliance.

From all available game data, 30 sequences of local game world states were extracted. Each sequence tracks a small set of players from three different alliances, two of which belong to the same meta-alliance (indicated by a fact $meta\_alliance(a1, a2)$). On average, sequences consist of 25.8 interpretations, every interpretation contains 16.4 cities and 10.6 players, and there are 17.6 conquest events per sequence. The 30 extracted sequences constitute positive examples. A further 60 negative examples were obtained by giving the wrong meta-alliance information (i.e., $meta\_alliance(a1, a3)$ or $meta\_alliance(a2, a3)$).

We hand-coded a simple CPT-theory that encodes a few basic features that one would assume to be useful in such a task, such as whether two players in different alliances $a1$ and $a2$ attack each other (indicating $\neg meta\_alliance(a1, a2)$), or jointly attack a player from a third alliance (indicating $meta\_alliance(a1, a2)$). As an example, consider the following rule:

$$conq(C, P1) : 0.0061 \vee nil : 0.9939$$
$$\longleftarrow city(C, \_, \_, P2), player(P2, \_, A1), player(P1, \_, A2), \neg meta\_alliance(A1, A2),$$

which states that the player $P1$ attacks a city C of a player $P2$ who is not his alliance partner.

Such a CPT-theory can be used for classification as follows. Given a set of training sequences $\mathcal{D}$, we first split this set into positive sequences $\mathcal{D}_+$ and negative sequences $\mathcal{D}_-$. We then learn the parameters of two CPT-theories $\mathcal{T}_+$ and $\mathcal{T}_-$ on the sets $\mathcal{D}_+$ and $\mathcal{D}_-$ according to maximum likelihood using the EM algorithm presented in Sect. 3.4. Note that $\mathcal{T}_+$ and $\mathcal{T}_-$ both employ the simple rule set outlined above, and only differ in their parameter values. Given a new test sequence S, we then evaluate the likelihood of S under the positive and negative models, $P(S \mid \mathcal{T}_+)$ and $P(S \mid \mathcal{T}_-)$, and predict the class for which this likelihood is higher.

To evaluate the accuracy of CPT-L in the meta alliance classification task, we performed a 10-fold cross-validation, using the same folds as used in Karwath et al. (2008). Figure 9 compares the results obtained for CPT-L with those of the BOOSTEDREAL system. BOOSTEDREAL is a state-of-the-art system for classification of (relational) sequences by alignment, which uses a discriminative approach based on boosting the reward model used in the alignment algorithm (Karwath et al. 2008). Note that BOOSTEDREAL, in contrast to CPT-L, is not a generative model for sequences of interpretations, but rather a discriminative approach specifically tailored to classification problems. It is also significantly more complex, and the resulting models are harder to interpret, as the boosted reward function is represented as an ensemble of relational regression trees. Figure 9 shows that CPT-L, at 82.22% with standard deviation of 9.37, achieves a slightly lower accuracy than the best observed result for BOOSTEDREAL, although the difference is not significant assuming equal variances for CPT-L and BOOSTEDREAL. Overall, we can conclude from this experiment that even with the simple rule set used, CPT-L is able to learn a model that captures useful information about the positive and negative class, and achieves similar accuracies as other state-of-the-art sequence classification schemes. Learning a single model in this domain takes under 2 minutes, using the lifted inference technique described in Sect. 3.3.

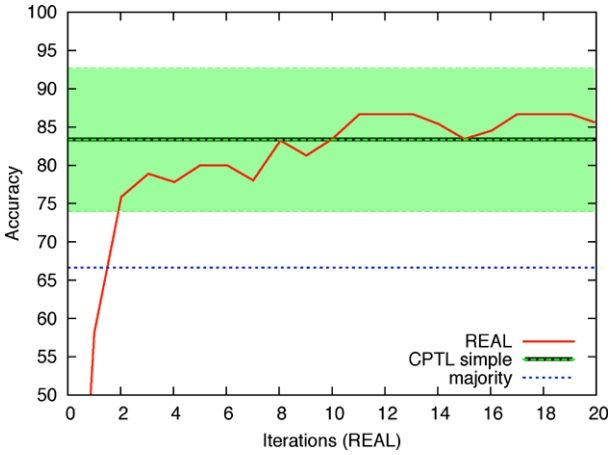

Fig. 9 Classification accuracy for the BOOSTEDREAL system (see Karwath et al. 2008) and CPT-L for the meta-alliance problem in the massively multiplayer online game domain. For BOOSTEDREAL, accuracy is a function of the boosting iteration (shown on the x-axis). For CPT-L, standard deviation over the cross-validation folds is indicated by the green-shaded area. Classification accuracy of the majority-class predictor is also shown

We are also currently trying to model this classification problem using discriminative Markov Logic Networks, in order to better understand the trade-offs between more general and simpler SRL approaches. However, with similar features as used in the CPT-L rules described above, we have not been able to obtain classification accuracies higher than majority class. The time needed for building a model in Markov Logic is approximate 2 hours, thus about two orders of magnitude higher than for our approach.

Prediction experiments We now consider the problem of predicting player actions within Travian, testing the prediction algorithm presented in Sect. 3.5. From all available data, we again extracted 30 sequences of local game world states. Each sequence involves a subset of 10 players, which are tracked over a period of one month (10 sequences each for December, January and February). Player sets are chosen such that there are no interactions between players in different sets, but a high number of interactions between players within one set. Cities that did not take part in any conquest event were removed from the data, leaving approximately 30–40 cities under consideration for every player subset.

We defined a world model in CPT-L that expresses the probability for player actions such as conquests of cities and changes in alliances affiliation, and updates the world state accordingly. Player actions in Travian—although strongly stochastic—are typically explainable from the social context of the game: different players from the same alliance jointly attack a certain territory on the map, there are retaliation attacks at the alliance level, or players leave alliances that have lost many cities in a short period of time. From a causal perspective, actions are thus triggered by certain (relational) patterns that hold in the game graph, which take into account a player's alliance affiliation together with the actions carried out by other alliance members. Such patterns can be naturally expressed in CPT-L as bodies of rules which trigger actions encoded in the head of the rule. We again manually defined a number of simple rules capturing such typical game patterns. As an example, consider the rules

$$conq(P, C) : 0.039 \vee nil : 0.961$$
$$\longleftarrow conq(P, C'), city(C', \_, \_, \_, P'), city(C, \_, \_, \_, P')$$
$$conq(P, C) : 0.011 \vee nil : 0.989$$
$$\longleftarrow city(C, \_, \_, \_, P''), allied(P, A), allied(P', A), conq(P', C'), city(C', \_, \_, \_, P'').$$

The first rule encodes that a player is likely to conquer a city of a player he or she already attacked in the previous time step. The second rule generalizes this pattern: a player P is likely to attack a city C of player P'' if an allied player has attacked P'' in the previous time step.

Moreover, the world state needs to be updated given the players' actions. After a conquest attack $conq(P, C)$, the city C changes ownership to player P in the next time step. If several players execute conquest attacks against the same city in one time step, one of them is chosen as the new owner of the city with uniform probability (note that such simultaneous conquest attacks would not be observed in the training data, as only one snapshot of the world is taken every 24 hours). Similarly, an $alliance\_change(P, A)$ event changes the alliance affiliation of player P to alliance A in the next time step.

We now consider the task of predicting the "conquest" action $conq(P, C)$ based on a learned model of world dynamics. The collected sequences of game states were split into one training set (sequences collected in December 2007) and two test sets (sequences collected in January 2008 and sequences collected in February 2008). Maximum-likelihood parameters of a hand-crafted CPT-theory $\mathcal{T}$ as described above were learned on the training set using EM. Afterwards, the learned model was used to predict the player action $conq(P, C)$ on the test data in the following way. Let S denote a test sequence with states $I_0, \ldots, I_T$. For every $t_0 \in \{0, \ldots, T-1\}$, and every player p and city c occurring in S, the learned model is used to compute the probability that the conquest event $conq(p, c)$ will be observed in the next world state, $P(I_{t_0+1} \models conq(p, c) \mid \mathcal{T}, I_0, \ldots, I_{t_0})$. This probability is obtained from the sampling-based prediction algorithm described in Sect. 3.5. The prediction is compared to the known ground truth (whether the conquest event occurred at that time in the game or not). Instead of predicting whether the player action will be taken in the next step, we can also predict whether it will be taken within the next k steps, by computing

$$P(I_{t_0+1} \models conq(p, c) \vee \cdots \vee I_{t_0+k} \models conq(p, c) \mid \mathcal{T}, I_0, \ldots, I_{t_0}).$$

This quantity is also easily obtained from the prediction algorithm described in Sect. 3.5.

Figure 10, left, shows ROC curves for this experiment with different values $k \in \{1, 2, 3, 4, 5\}$, evaluated on the first test set (January 2008). Figure 10, right, shows the corresponding AUC values as a function of k for both test sets. The achieved area under the ROC curve is substantially above 0.5 (random performance), indicating that the learned

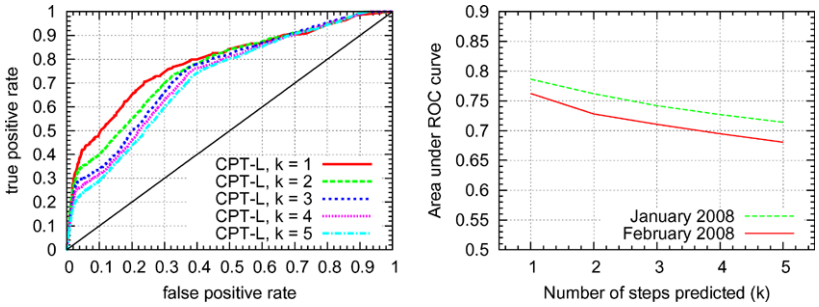

Fig. 10 *Left figure*: ROC curve for predicting that a city C will be conquered by a player P within the next k time steps, for $k \in \{1, 2, 3, 4, 5\}$. The model was trained on 10 sequences of local game state descriptions from December 2007, and tested on 10 sequences from January 2008. *Right figure*: AUC as a function of the number k of future time steps considered in the same experiment. Additionally, AUC as a function of k is shown for 10 test sequences from February 2008

CPT-theory $\mathcal{T}$ indeed captures some characteristics of player behavior and obtains a reasonable ranking of player/city pairs (p/c) according to the probability that p will conquer c. Moreover, the model is able to predict conquest actions several steps in the future, although AUC is slightly lower for larger k. This indicates that uncertainty associated with predictions accumulates over time. Finally, predictions for the first test set (January 2008) are slightly more accurate than for the second test set (February 2008). This is not surprising as the model has been trained from sequences collected in December 2007, and indicates a slight change in game dynamics over time. In summary, we conclude that player actions in Travian are indeed to some degree predictable from the social context of the game, and CPT-L is able to learn such patterns from the data. The computational complexity of learning in this task will be analyzed in detail in the next section.

Scaling experiments We now analyze the scaling behavior of the proposed algorithms in detail, and compare the basic inference algorithm presented in Sect. 3.2 to the lifted inference algorithm presented in Sect. 3.3. To this end, we again consider the prediction setting discussed in the last section, and vary the number of players and cities that are present in any given game state. We used data containing up to 50 players, which together controlled up to 269 cities. As before, 30 sequences of such game states were extracted from the data. To evaluate computational complexity, a model was trained on all sequences, using the same rule set as used for the prediction task.

To illustrate the complexity of the resulting problem, one can approximate the size of the ground network that would have been obtained had we grounded the model to a Bayesian or Markov Network as it is typically done for SRL approaches such as CP-logic or Markov Logic Networks. In such a network, nodes correspond to all groundings of predicates using available domain constants. Note that in general, only the part of this network that is relevant for a particular query needs to be constructed. However, in our scenario all ground facts involving constants that appear in a training sequence are relevant when learning from that sequence or computing its likelihood. Furthermore, in the dynamic setting considered here, the network has to be unrolled over time, essentially duplicating the nodes for every time step in the observation sequence. For the largest domain we have considered (involving 50 players and 269 cities), the size of the ground network is approximately 800.000 nodes, indicating that exact inference and learning in this network would be computationally expensive.

Figure 11 shows the time needed to perform inference in CPT-L in the outlined domain as a function of the size of the (hypothetical) ground network, for up to 20 players. Timing results are given for both the basic inference algorithm presented in Sect. 3.2 and the lifted inference algorithm presented in Sect. 3.3. It can be observed that the lifted inference algorithm has significantly better scaling behavior, and achieves a speed-up of about a factor of 50 compared to the basic inference algorithm in large domains. For datasets containing more than 20 players, the standard inference algorithm could not be run anymore. However, we ran the lifted inference algorithm for datasets with up to 50 players, resulting in the (hypothetical) ground networks of approximately 800.000 nodes mentioned above. In this setting, lifted inference could still be performed in about 2 seconds.

Overall, these experiments show that the introduced simple lifted inference algorithm yields a substantial speed-up compared to the basic inference algorithm. Note that the inference we perform is exact, and computational efficiency is achieved by exploiting the relative simplicity of our model and learning setting. This is in contrast to other approaches that try to overcome the excessive size of ground networks by performing approximate inference, as, for example, in Markov Logic Networks (Richardson and Domingos 2006).

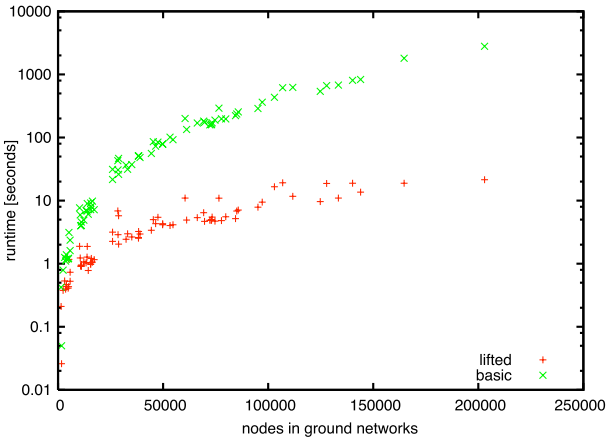

Fig. 11 Time for performing inference (in the Expectation Step of the EM algorithm) for the Travian prediction task as a function of the domain size. The y-axis shows runtime in seconds. The x-axis shows the number of nodes in the Bayesian network that would result from the grounding of the CPT-theory in this domain

5 Related work

There are relatively few existing approaches that can probabilistically model sequences of relational state descriptions. CPT-L can be positioned with respect to them as follows.

First, statistical relational learning systems such as Markov Logic (Richardson and Domingos 2006), CP-logic (Vennekens et al. 2006), Probabilistic Relational Models (Getoor et al. 2001) or Bayesian Logic Programs (Kersting and De Raedt 2007) can be used in this setting by adding an extra time argument to predicates (then called *fluents*). However, inference and learning in these systems is computationally expensive: they support very general models including hidden states, and are not optimized for sequential data. A second class of techniques, for instance (Zettlemoyer et al. 2005), uses transition models based on (stochastic) STRIPS rules. This somewhat limits the transitions that can be expressed, as only one rule "fires" at every point in time, and it is difficult to model several processes that change the state of the world concurrently (such as an agent's actions and naturally occurring world changes). Related to this, is the probabilistic extension of PPDDL (Younes and Littman 2004) that has been developed for the ICAPS planning competition and that form a generalization of STRIPS. From a representational perspective, PPDDL is equivalent to Dynamic Bayesian nets as actions in PPDDL are restricted by finite domains. PPDDL also employs frame axioms. Writing PPDDL is, however, difficult because the user is supposed to ensure that the theory is consistent and, hence, that consistency is not enforced by the language. This makes significantly complicates structure learning for PPDDL models. Note that it is very well possible that the algorithms presented in this paper can be adapted towards PPDDL and this seem interesting direction for further research.

Another related formalism is that of Logical MDPs (Kersting and De Raedt 2003), which specifically targets Markov Decision Processes and thus takes into account rewards. The action rules employed in LoMDPs are somewhat similar to CPT-L rules, but they require that the bodies of the action rules are mutually exclusive (which is achieved by imposing an order on the rules). CP-logic, and therefore also CPT-L, does neither impose orders on rules, nor does it require that only one clause triggers at the same time, which makes it more natural to model stochastic relational processes.

Another approach designed to model sequences of relational state descriptions are relational simple-transition models (Fern 2005). A related approach is that by Biswas et al. (2007), who employs dynamic Markov Logic to represent stochastic relational processes.

Inference is carried out in a ground dynamic Bayesian network constructed from the MLN. In contrast to CPT-L, these two approaches focus on domains where the process generating the data is hidden, and inferring these hidden states from observations. This is a significantly harder setting than the fully observable setting discussed in this paper, and therefore typically only approximate inference is possible (Fern 2005). However, we feel that also the easier problem where everything is observable is worthy of investigation in its own right. A better understanding of this problem should also provide new insights into the more complex one. In this context, we can mention that an extension of CPT-L to deal with hidden variables is currently under study, where inference is based on a Monte Carlo method, cf. Thon (2009).

6 Conclusions and future work

We have introduced CPT-L, a probabilistic model for sequences of relational state descriptions. In contrast to other approaches that could be used as a model for such sequences, CPT-L focuses on computational efficiency rather than expressivity. We have specifically discussed how to perform efficient inference and parameter learning in CPT-L by a partially lifted inference algorithm. The algorithm aggregates all groundings of rules where the chosen head element is logically entailed into a joint factor during probabilistic inference, thereby significantly reducing the size of the resulting inference problem. We have also extended earlier work on CPT-L by relaxing the Markov assumption on the underlying stochastic process, and using more flexible rules where rule heads consist of a disjunction of conjunctions.

There are two main directions for future work. One direction is structural optimization, that is, learning entire rule sets from data as opposed to only learning parameters for a given rule set. We are currently trying to infer rules for CPT-L using standard rule learners such as Progol or Tertius. Experiments in this direction are promising but preliminary. A second interesting direction for future work is to extend the model towards a setting where data is only partially observed. We have started studying an extension of CPT-L in which a subset of domain predicates is hidden, while other predicates have to be fully observable. The resulting hidden state inference problem is computationally challenging, thus it likely calls for approximate inference techniques. Some initial encouraging results were achieved in this setting using particle filters (Thon 2009). Finally, we are interested in applying the presented techniques in other challenging application domains.

Appendix A: Proof of Theorem 2

Theorem 2 *Let $\mathcal{B}$ be a BDD resulting from the conversion of an inference problem $P(I_{t+1} \mid I_{[0,t]})$, annotated with upward and downward probabilities as defined above, and let*

$$\Gamma = \{\sigma \mid I_{[0,t]} \xrightarrow{\sigma} I_{t+1}\}$$

be the set of selections yielding I_{t+1}. Then

$$\begin{aligned} \alpha(root) &= \sum_{\sigma \in \Gamma} P(\sigma) \\ &= P(I_{t+1} \mid I_{[0,t]}). \end{aligned} \tag{7}$$

Proof Let $\mathcal{B}$ be a BDD graph structure resulting from an inference problem $p(I_{t+1} \mid I_{[0,t]})$, and let the nodes in $\mathcal{B}$ be annotated with upward and downward probabilities as outlined in Sect. 3.2. Let $\mathcal{N}$ and $\mathcal{E}$ denote the nodes and edges in $\mathcal{B}$. To every edge $E \in \mathcal{E}$ we associate a weight $P(E)$ with

$$P(E) = \begin{cases} P(c \mid r), & E \text{ corresponds to a positive branch} \\ 1, & E \text{ corresponds to a negative branch} \end{cases}$$

where $r.c$ is the Boolean variable associated with the node N from which E originates, and $P(c \mid r)$ the probability of choosing head element c in rule r. A (directed) path R in $\mathcal{B}$ is a sequence $N_1 E_1 \ldots N_k E_k N_{k+1}$ with $E_i \in \mathcal{E}$ and $N_i \in \mathcal{N}$, and we always go downward in the BDD. We define the weight of a path as

$$P(R) = \prod_{i=1}^{k} P(E_i), \tag{16}$$

and denote by $\mathcal{R}(N)$ the set of all paths from a node $N \in \mathcal{N}$ to the 1-terminal. We first show that

$$\forall N \in \mathcal{N} : \alpha(N) = \sum_{R \in \mathcal{R}(N)} P(R), \tag{17}$$

by induction over the level of a node in $\mathcal{B}$.

Base Case: We need to show (17) for the terminal nodes. If N is the 1-terminal, the (trivial) path $R = N$ is the only element of $\mathcal{R}(N)$, with $P(R) = 1$ according to (16). Thus, (17) holds. If N is the 0-terminal node, then $\mathcal{R}(N) = \emptyset$, thus $\sum_{R \in \mathcal{R}(N)} P(R) = 0$, and (17) holds as well.

Induction: Let $N \in \mathcal{N}$ denote a non-terminal node, and let $r.c$ denote its associated Boolean variable. Let E^+ and E^- denote the positive and negative branch originating from N, and N^+ and N^- the corresponding child nodes. A path $R \in \mathcal{R}(N)$ either runs through E^+ or E^-. In the first case, we have $R = N E^+ R'$ with $R' \in \mathcal{R}(N^+)$, and $P(R) = P(c \mid h) P(R')$. In the second case, we have $R = N E^- R'$ with $R' \in \mathcal{R}(N^-)$, and $P(R) = P(R')$. Thus,

$$\sum_{R \in \mathcal{R}(N)} P(R) = P(c \mid r) \sum_{R \in \mathcal{R}(N^+)} P(R) + \sum_{R \in \mathcal{R}(N^-)} P(R).$$

From the inductive assumption it follows that

$$\begin{aligned} \sum_{R \in \mathcal{R}(N)} P(R) &= P(c \mid r) \alpha(N^+) + \alpha(N^-) \\ &= \alpha(N), \end{aligned}$$

completing the proof of (17). Recall that there is a one-to-one correspondence between a selection σ yielding I_{t+1} and a path R from the root to the 1-terminal. Considering (1) and (16), we also see that $P(\sigma) = P(R)$. Thus,

$$\begin{aligned} \alpha(root) &= \sum_{R \in \mathcal{R}(root)} P(R) \\ &= \sum_{\sigma \in \Gamma} P(\sigma) \\ &= P(I_{t+1} \mid I_{[0,t]}), \end{aligned}$$

completing the proof of Theorem 2. □

Appendix B: Proof of Theorem 3

Before proving Theorem 3 we will prove the following lemma:

Lemma 1 *Let $\mathcal{B}$ be a BDD graph structure resulting from an inference problem $p(I_{t+1} \mid I_{[0,t]})$, let the nodes in $\mathcal{B}$ be annotated with upward and downward probabilities as outlined in Sect. 3.2, and let $N_1, \ldots, N_k$ denote all nodes at a given level n in the BDD. Then it holds that*

$$P(I_{t+1} \mid I_{[0,t]}) = \sum_{l=1}^{k} \beta(N_l)\alpha(N_l). \tag{18}$$

Proof We prove Lemma 1 by induction over the BDD level n.

Base case: $n = 0$. At level zero of the BDD, there is only a single node, namely the root node. Equation (18) follows from Theorem 2 and $\beta(root) = 1$ (5):

$$\alpha(root) = \alpha(root)\beta(root) = P(I_{t+1} \mid I_{[0,t]}).$$

Induction: Assume that (18) holds for level n. Let $r_i\theta.c_{ij}\theta$ denote the Boolean variable associated with level n in the BDD, and let $p = P(r_i.c_{ij})$ be the corresponding probability. Let N_l with $l = 1, \ldots, k$ denote all nodes at level n, and let N'_l with $l = 1, \ldots, k'$ denote all nodes at level $n+1$. Let furthermore N_l^+ (N_l^-) denote the positive (negative) child node of N_l for $l = 1, \ldots, k$. We will refer by $pa^+(N'_l)$ ($pa^-(N'_l)$) to the subset of the nodes $N_1, \ldots, N_k$ which have N'_l as positive (negative) child node.

Starting from the inductive assumption, we now derive

$$\begin{aligned}
P(I_{t+1} \mid I_{[0,t]}) &= \sum_{l=1}^{k} \alpha(N_l)\beta(N_l) \\
&= \sum_{l=1}^{k} \alpha(N_l^+)\beta(N_l)p + \sum_{l=1}^{k} \alpha(N_l^-)\beta(N_l) && (19) \\
&= \sum_{l'=1}^{k'} \Bigg[\sum_{N_l \in pa^+(N'_{l'})} \alpha(N'_{l'})\beta(N_l)p + \sum_{N_l \in pa^-(N'_{l'})} \alpha(N'_{l'})\beta(N_l) \Bigg] && (20) \\
&= \sum_{l'=1}^{k'} \alpha(N'_{l'})\beta(N'_{l'}). && (21)
\end{aligned}$$

Equation (19) follows from the definition of upward probabilities α. To derive (20), we note that each edge from a node at level n either goes to a node at level $n+1$, or to the 0-terminal; because $\alpha(\text{zero-terminal}) = 0$ the sums in (19) and (20) thus contain the same terms (see also Fig. 12). Finally, (21) follows from the definition of downward probabilities β. □

Theorem 3 *Let p_{ij} be the parameter associated with head element c_{ij} in rule r_i, let $\Delta = I_{[0,t]} \to I_{t+1}$ be a single transition, and let $r_i\theta \in \mathbf{R}_t$ denote a grounding of r_i applicable in $I_{[0,t]}$. Let $N_1, \ldots, N_k$ be all nodes in the BDD associated with the Boolean variable $r_i\theta.c_{ij}\theta$*

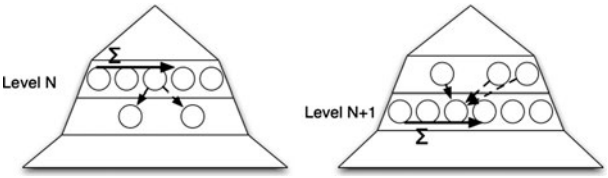

Fig. 12 Inductive step in proof of $P(I_{t+1} \mid I_{[0,t]}) = \sum_{\sigma \in \Gamma} P(\sigma) = \sum_{l=1}^{k} \beta(N_l)\alpha(N_l)$

resulting from the grounded rule $r_i\theta$, and let N_l^+ be the child on the positive branch of N_l. Then

$$\mathbb{E}[\kappa_{ij}^{\theta} \mid \pi, \Delta] = \frac{1}{P(I_{t+1} \mid I_{[0,t]})} \sum_{l=1}^{k} \beta(N_l) p_{ij} \alpha(N_l^+). \tag{15}$$

Proof The nodes $N_1, \ldots, N_k$ associated with the variable $r_i\theta.c_{ij}\theta$ together form a level n of the BDD. As above let N_l^- denote the child on the negative branch of node N_l. Reconsidering (19) in the proof of Lemma 1, we see that the probability of head element c_{ij} being selected is given by

$$P(\kappa_{ij}^{\theta} = 1 \mid \pi, \Delta) = \frac{\sum_{l=1}^{k} \alpha(N_l^+)\beta(N_l)p}{\sum_{l=1}^{k} \alpha(N_l^+)\beta(N_l)p + \sum_{l=1}^{k} \alpha(N_l^-)\beta(N_l)} \tag{22}$$

as the head element is chosen if and only if a node at level n is left through the positive branch. Because κ_{ij}^{θ} is a binary indicator,

$$\begin{aligned} \mathbb{E}[\kappa_{ij}^{\theta} \mid \pi, \Delta] &= P(\kappa_{ij}^{\theta} = 1 \mid \pi, \Delta) \\ &= \frac{\sum_{l=1}^{k} \alpha(N_l^+)\beta(N_l)p}{\sum_{l=1}^{k} \alpha(N_l^+)\beta(N_l)p + \sum_{l=1}^{k} \alpha(N_l^-)\beta(N_l)} \\ &= \frac{1}{P(I_{t+1} \mid I_{[0,t]})} \sum_{l=1}^{k} \alpha(N_l^+)\beta(N_l)p \end{aligned} \tag{23}$$

where (23) follows from the definition of downward probabilities β and Lemma 1. □

References

Biswas, R., Thrun, S., & Fujimura, K. (2007). Recognizing activities with multiple cues. In *Lecture notes in computer science* (Vol. 4814, p. 255). Berlin: Springer.

Bratko, I. (1990). *Prolog programming for artificial intelligence* (2nd edn.). Reading: Addison-Wesley.

Bryant, R. E. (1986). Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, *35*(8), 677–691.

De Raedt, L., Kimmig, A., & Toivonen, H. (2007). ProbLog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th international joint conference on artificial intelligence* (pp. 2462–2467).

De Raedt, L., Frasconi, P., Kersting, K., & Muggleton, S. (Eds.) (2008). *Lecture notes in computer science*: *Vol. 4911. Probabilistic inductive logic programming—theory and applications*. Berlin: Springer.

Fern, A. (2005). A simple-transition model for relational sequences. In *Proceedings of the 19th international joint conference on artificial intelligence* (pp. 696–701). Edinburgh, Scotland, UK.

Fikes, R. E., & Nilsson, N. J. (1995). STRIPS: a new approach to the application of theorem proving to problem solving. In *Computation & intelligence: collected readings* (pp. 429–446). Menlo Park, CA, USA, American Association for Artificial Intelligence.

Getoor, L., & Taskar, B. (Eds.) (2007). *Statistical relational learning*. Cambridge: MIT Press.
Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. In *Relational data mining* (pp. 307–335). Berlin: Springer.
Ghahramani, Z. (1997). Learning dynamic Bayesian networks. In *Adaptive processing of sequences and data structures* (pp. 168–197). International summer school on neural networks.
Jaeger, M. (1997). Relational Bayesian networks. In D. Geiger & P. Shenoy (Eds.), *Proceedings of the thirteenth annual conference on uncertainty in artificial intelligence (UAI-97)* (pp. 266–273). Providence, Rhode Island, USA, San Mateo: Morgan Kaufmann.
Karwath, A., Kersting, K., & Landwehr, N. (2008). Boosting relational sequence alignments. In: *Proceedings of the 8th IEEE international conference on data mining (ICDM 2008)*.
Kersting, K., & De Raedt, L. (2003). Logical Markov decision programs. In *Proceedings of the international joint conference on artificial intelligence IJCAI-2003* (Vol. 3).
Kersting, K., & De Raedt, L. (2007). Bayesian logic programming: theory and tool. In L. Getoor & B. Taskar (Eds.), *An introduction to statistical relational learning*. Cambridge: MIT Press.
Laird, J. E., & van Lent, M. (2000). Human-Level AI's killer application: Interactive computer games. In *Proceedings of the seventeenth national conference on artificial intelligence and twelfth conference on innovative applications of artificial intelligence*.
Milch, B., Zettlemoyer, L. S., Kersting, K., Haimes, M., & Kaelbling, L. P. (2008). Lifted probabilistic inference with counting formulas. In *Proceedings of the 23rd national conference on artificial intelligence (AAAI-2008)*.
Mutton, P. (2004). Inferring and visualizing social networks on Internet Relay Chat. In *Proceedings of the 8th international conference on information visualization (IV-2004)* (pp. 35–43).
Pollack, M. E. (2005). Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. *AI Magazine*, *26*(2), 9–24.
Poole, D. (1997). The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, *94*(1–2), 7–56.
Poole, D. (2003). First-order probabilistic inference. In G. Gottlob & T. Walsh (Eds.), *IJCAI* (pp. 985–991). San Mateo: Morgan Kaufmann.
Puterman, M. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. New York: Wiley.
Rabiner, L. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.
Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.
Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. In *Proceedings of the twelfth international conference on logic programming (ICLP-1995)* (pp. 715–729).
Sato, T., & Kameya, Y. (1997). PRISM: A symbolic-statistical modeling language. In *Proceedings of the 15th international joint conference on artificial intelligence (IJCAI-97)* (pp. 1330–1339). San Mateo: Morgan Kaufmann.
Saul, L. K., & Jordan, M. I. (1999). Mixed memory Markov models: Decomposing complex stochastic processes as mixtures of simpler ones. *Machine Learning*, *37*, 75–87.
Thon, I. (2009). Don't fear optimality: sampling for probabilistic-logic sequence models. In *Proceedings of the international conference on inductive logic programming (ILP-2009)*, July 2009.
Thon, I., Landwehr, N., & Raedt, L. D. (2008). A simple model for sequences of relational state descriptions. In *Proceedings of the 19th European conference on machine learning* (pp. 506–521).
Thon, I., Gutmann, B., van Otterlo, M., Landwehr, N., & Raedt, L. D. (2009). From non-deterministic to probabilistic planning with the help of statistical relational learning. In *ICAPS 2009—Proceedings of the workshop on planning and learning*, September 2009.
Vennekens, J., Denecker, M., & Bruynooghe, M. (2006). Representing causal information about a probabilistic process. In *Lecture Notes in Computer Science: Vol. 4160. Logics in artificial intelligence* (pp. 452–464). Berlin: Springer.
Younes, H. L., & Littman, M. L. (2004). PPDDL1.0: The language for the probabilistic Part of IPC-4. In *Proceedings of the international planning competition*.
Zettlemoyer, L. S., Pasula, H., & Kaelbling, L. P. (2005). Learning planning rules in noisy stochastic worlds. In *Proceedings of the 20th national conference on artificial intelligence (AAAI-05)* (pp. 911–918).