A Simple Model for Sequences of Relational State Descriptions

Ingo Thon, Niels Landwehr, and Luc De Raedt

Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Heverlee, Belgium
{firstname.lastname}@cs.kuleuven.be

Abstract. Artificial intelligence aims at developing 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. Standard probabilistic sequence models provide efficient inference and learning techniques, but typically cannot fully capture the relational complexity. On the other hand, statistical relational learning techniques are often too inefficient. In this paper, we present a simple model that occupies an intermediate position in this expressiveness/efficiency trade-off. It is based on CP-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. We show that the resulting model is able to handle probabilistic relational domains with a substantial number of objects and relations.

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 [1]. Such a model could 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 multi-player 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 test-bed for developing and evaluating artificial intelligence techniques and are also interesting in their own right (cf. also [2]). 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 in this type of world can be useful in several ways. Analysis of in-game social networks are 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

W. Daelemans et al. (Eds.): ECML PKDD 2008, Part II, LNAI 5212, pp. 506–521, 2008.

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 todays 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) world state descriptions are inherently relational, as the interaction between (groups of) agents is of central interest, 2) the transition behavior of the world will be 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.

Artificial intelligence has already contributed a rich variety of different modeling approaches, for instance, Markov models [3] and decision processes [4], dynamic Bayesian networks [5], STRIPS [6], statistical relational learning representations [7], etc. Most of the existing approaches that support reasoning about uncertainty (and satisfy requirement 2) employ essentially propositional representations (for instance, dynamic Bayesian networks, Markov models, etc.), and are not able to represent complex relational worlds, and hence, 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, CP-logic, or BLPs) is considered within statistical relational learning [7] and probabilistic inductive logic programming [8]. However, the inefficiency of inference and learning algorithms causes problems in many realistic applications, and hence, such methods do not satisfy requirement 3).

We aim to alleviate this situation, by contributing a novel representation, called CPT-L (for **CPT**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 [9], 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 its predecessor and alternative formalisms within the artificial intelligence and statistical relational learning literature.

This paper is organized as follows: Section 2 introduces the CPT-L framework; Section 3 addresses inference and parameter estimation, while Section 4 presents some experimental results in both the block's world and the MMOG Travian. Finally, Section 5 discusses related work, before concluding and touching upon future work in Section 6.

2 CPT-L

Let us first introduce some terminology. An *atom* is an expression of the form $p(t_1, ..., t_n)$ where p/n is a *predicate symbol* and the t_i are *terms*. Terms are built up from constants, variables and functor symbols. 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. We are interested in describing complex world states in terms of *relational interpretations*. A relational interpretation I is a set of ground facts $a_1, ..., a_N$. A *relational stochastic process* defines a distribution $P(I_0, ..., I_T)$ over sequences of interpretations of length T, and thereby completely characterizes the transition behavior of the world.

The semantics of CPT-L is based on CP-logic, a probabilistic first-order logic that defines probability distributions over interpretations [9]. CP-logic has a strong focus on causality and constructive processes: an interpretation is incrementally constructed by a process that adds facts which are probabilistic *outcomes* of other already given facts (the *causes*). CPT-L combines the semantics of CP-logic with that of (first-order) Markov processes. Causal influences only stretch from I_t to I_{t+1} (Markov assumption), are identical for all time-steps (stationarity), and all causes and outcomes are observable. Models in CPT-L are also called CP-theories, and can be formally defined as follows:

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

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

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 $b_1, ..., b_m$ as the body $body(r)$ of the rule and to $(h_1 : p_1) \vee \ldots \vee (h_n : p_n)$ as the head $head(r)$ of the rule. 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. Rules define conditional probabilistic events: the intuition behind a rule is that whenever $b_1\theta, ..., b_m\theta$ holds for a substitution θ in the current state I_t, exactly one of the $h_i\theta$ in the head will hold in the next state I_{t+1}. In this way, the rule models a (probabilistic) causal process as the condition specified in the body causes one (probabilistically chosen) atom in the head to become true in the next time-step.

Example 1. Consider the following CPT-rule:

$$(on(A, table) : 0.9) \vee (on(A, C) : 0.1) \leftarrow free(A), on(A, C), move(A, table).$$

which represents that we try to move a block A from block C to the table. This action succeeds with a probability of 0.9.

We now show how a CPT-theory defines a distribution over sequences $I_0, ..., I_T$ of relational interpretations. Let us first define the concept of the applicable ground rules in an interpretation I_t. From a CPT-theory, the rule $(h_1 : p_1\theta) \vee \ldots \vee (h_n : p_n\theta) \leftarrow b_1\theta, \ldots, b_m\theta$ is obtained for a substitution θ. A ground rule r is applicable in I_t if and only if $body(r) = b_1\theta, \ldots, b_m\theta$ is true in I_t, denoted $I_t \models b_1\theta, \ldots, b_m\theta$.

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, cf. [10]. In the presence of background knowledge, a ground rule is applicable in an interpretation I_t if $b_1\theta, \ldots, b_m\theta$ can be logically derived from I_t together with the logic program B, denoted $I_t \models_B b_1\theta, \ldots, b_m\theta$.

The set of all applicable ground rules in state I_t will be denoted as $\mathbf{R}_t$. Each ground rule applicable in I_t will cause one of its head elements to become true in I_{t+1}. More formally, let $\mathbf{R}_t = \{r_1, ..., r_k\}$. A *selection* σ is a mapping $\{(r_1, j_1), ..., (r_k, j_k)\}$ from ground rules r_i to indices j_i denoting that head element $h_{ij_i} \in head(r_i)$ is selected. The probability of a selection σ is

$$P(\sigma) = \prod_{i=1}^{k} p_{j_i}, \tag{1}$$

where p_{j_i} is the probability associated with head element h_{ij_i} in r_i. In the stochastic process to be defined, I_{t+1} is a possible successor for the state I_t if and only if there is a selection σ such that $I_{t+1} = \{h_{1\sigma(1)}, ..., h_{k\sigma(k)}\}$. We shall say that σ *yields* I_{t+1} in I_t, denoted $I_t \xrightarrow{\sigma} I_{t+1}$, and define

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

Example 2. Consider the theory

$$\begin{aligned} r_1 &= a : 0.2 \vee b : 0.8 \leftarrow \neg a, \neg b \\ r_2 &= a : 0.5 \vee b : 0.5 \leftarrow a \\ r_3 &= a : 0.7 \vee nil : 0.3 \leftarrow a \end{aligned}$$

Starting from $I_t = \{a\}$ only the rules r_2 and r_3 are applicable, so $\mathbf{R}_t = \{r_2, r_3\}$. The set of possible selections is

$$\{(r_2, j_2), (r_3, j_3) \mid j_2, j_3 \in \{1, 2\}\}.$$

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

$$\begin{aligned} &I^1_{t+1} = \{a\} \text{ with } P(I^1_{t+1} \mid I_t) = 0.5 \cdot 0.7 + 0.5 \cdot 0.3 = 0.5 \\ &I^2_{t+1} = \{b\} \text{ with } P(I^2_{t+1} \mid I_t) = 0.5 \cdot 0.3 = 0.15 \\ &I^3_{t+1} = \{a, b\} \text{ with } P(I^3_{t+1} \mid I_t) = 0.5 \cdot 0.7 = 0.35 \end{aligned}$$

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

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

Intuitively, it is clear that this defines a distribution over all sequences of interpretations of length T much as in the propositional case. More formally:

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 $P(I_0, ..., I_T)$ over sequences of interpretations of length T.*

3 Inference and Parameter Estimation in CPT-L

As for other probabilistic models, we can now ask several questions about the introduced CPT-L model:

- **Sampling:** How to sample sequences of interpretations $I_0, ..., I_T$ from a given CPT-theory $\mathcal{T}$ and initial interpretation I_0?
- **Inference:** Given a CPT-theory $\mathcal{T}$ and a sequence of interpretations $I_0, ..., I_T$, what is $P(I_0, ..., I_T \mid \mathcal{T})$?
- **Parameter Estimation:** Given the structure of a CPT-theory $\mathcal{T}$ and a set of sequences of interpretations, what are the maximum-likelihood parameters of $\mathcal{T}$?
- **Prediction:** Let $\mathcal{T}$ be a CPT-theory, $I_0, ..., I_t$ a sequence of interpretations, and F a first-order formula that constitutes a certain property of interest. What is the probability that F holds at time $t + d$, $P(I_{t+d} \models_B F \mid \mathcal{T}, I_0, ..., I_t)$?

Sampling from a CPT-theory $\mathcal{T}$ given an initial interpretation I_0 is straightforward due to the causal semantics employed in CP-logic. For $t \geq 0$, I_{t+1} can be constructed from I_t by finding all groundings $r\theta$ of rules $r \in \mathcal{T}$, and sampling for each $r\theta$ a head element to be added to I_{t+1}. Algorithmic solutions for solving the inference, parameter estimation, and prediction problem will be presented in turn in the rest of this section.

3.1 Inference

Because of the Markov assumption (Equation 3), the crucial task for solving the inference problem is to compute $P(I_{t+1} \mid I_t)$ for given I_{t+1} and I_t. According to Equation 2, this involves summing the probabilities of all selections yielding I_{t+1} from I_t. However, the number of possible selections σ is exponential in the number of ground rules $|\mathbf{R}_t|$ applicable in I_t, so a naive generate-and-test approach is infeasible. Instead, we present an efficient approach for computing $P(I_{t+1} \mid I_t)$ without explicitly enumerating all selections yielding I_{t+1}, which is strongly related to the inference technique discussed in [11]. The problem is first converted to a DNF formula over boolean variables such that assignments to variables correspond to selections, and satisfying assignments to selections yielding I_{t+1}. The formula is then compactly represented as a binary decision diagram (BDD), and $P(I_{t+1} \mid I_t)$ efficiently computed from the BDD using dynamic programming. Although finding satisfying assignments for DNF 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 a given inference problem to a DNF formula f is realized as follows:

1. Initialize $f := true$
2. Compute applicable ground rules

$$\mathbf{R}_t = \{r\theta | body(r\theta) \text{ is true in } I_t\}$$

3. For all rules $(r = (p_1 : h_1, ..., p_n : h_n) \leftarrow b_1, ..., b_m)$ in $\mathbf{R}_t$ do:
   (a) $f := f \wedge (r.h_1 \vee ... \vee r.h_n)$, where $r.h$ denotes the proposition obtained by concatenating the name of the rule r with the ground literal h resulting in a new propositional variable $r.h$ (if not $h_i = nil$).
   (b) $f := f \wedge (\neg r.h_i \vee \neg r.h_j)$ for all $i \neq j$

4. For all facts $l \in I_{t+1}$
   (a) Initialize $g := false$
   (b) for all $r \in \mathbf{R}_t$ with $p : l \in head(r)$ do $g := g \vee r.l$
   (c) $f := f \wedge g$

Boolean variables of the form $r.h$ represent that head element h was selected in rule r[1]. The second step of the algorithm computes all applicable rules, the third step assures that selections are obtained, and the final step assures that the selection generates the interpretation I_{t+1}. It is easily verified that the satisfying assignments for the formula f correspond to the selections yielding I_{t+1}.

Example 3. The following formula f is obtained for the transition $\{a\} \rightarrow \{a, b\}$ and the CPT-theory given in Example 2.

$$\underbrace{(r2.a \vee r2.b)}_{3.a} \wedge \underbrace{(\neg r2.a \vee \neg r2.b) \wedge (\neg r3.a \vee \neg r3.nil)}_{3.b} \wedge \underbrace{(r2.a \vee r3.a) \wedge r2.b}_{4}$$

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) [12] is constructed. Let $x_1, ..., x_n$ denote an ordered set of boolean variables (such as the $r.h$ 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 $x_1, ..., x_n$: given an instantiation of the x_i, we follow a path from the root to either 1 or 0 (indicating 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 (cf. [12] for details). Figure 1, left, shows an example BDD.

From the BDD graph, $P(I_{t+1} \mid I_t)$ can be computed in linear time using dynamic programming. This is realized by a straightforward modification of the algorithm for inference in ProbLog theories [11]. The algorithm exploits that paths in the BDD from the root node to the 1-terminal correspond to satisfying assignments for f, and thus selections yielding I_{t+1}. By sweeping through the BDD from top to bottom contributions from all such selections are summed up (Equation 2) without explicitly enumerating all paths. The efficiency of this method crucially depends on the size of the BDD graph, which in turn depends strongly on the chosen variable ordering $x_1, ..., x_n$. Unfortunately, computing an optimal variable ordering is NP-hard. However, existing implementations of BDD packages contain sophisticated heuristics to find a good ordering for a given function in polynomial time.

Interestingly, it is possible to further reduce complexity for the particular problem we are interested in by adapting a different semantics in the BDD. A *zero-suppressed binary decision* diagrams (or ZDD) is an alternative form of graphical representation

[1] Variables $r.h$ are standardized apart in case head elements coincide after grounding.

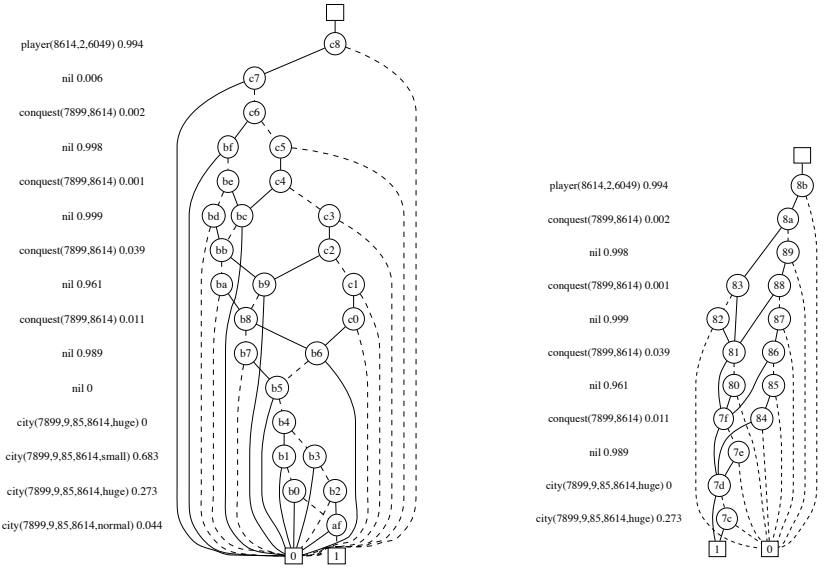

Fig. 1. Graphical representation of a formula f resulting from the conversion of a CPT-L inference problem represented as a BDD (left) and ZDD (right)

in which variables appear in a path only if their positive branch is not directly connected to the terminal 0 [13]. Figure 1 shows example BDD and ZDD structures that represent the same function. We will now show that a reduced ZDD representation of f will always be smaller than (or identical to) the corresponding BDD representation for CPT-L:

Theorem 2. *Let f be a formula resulting from the conversion of a CPT-L inference problem, G its BDD representation, and G' its ZDD representation (for a fixed variable ordering). Then $size(G') \leq size(G)$.*

Proof. We first show that in G every path Q from the root to the 1-terminal contains all variables appearing in f. Assume $r.h_1 \notin Q$, and let $r.h_2, ..., r.h_l$ denote the variables corresponding to the other head elements of rule r. Because of the constraint added in step 3. of the conversion, f can only be true if exactly one of the $r.h_1, ..., r.h_l$ is true. However, this cannot be verified by looking at any subset of the variables, and therefore they must all be contained in the path. Because all variables appear in every path from the root to 1, the graph structure G is also a faithful representation of f under the ZDD semantics. If G as a ZDD is fully reduced, $G = G'$ because reduced ZDDs, as BDDs, are a canonical representation. Otherwise, G can be further reduced to the ZDD G' with $size(G') < size(G)$.

Typically a ZDD representation of f will be more compact than the BDD representation, as shown in Figure 1.

3.2 Parameter Estimation

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

$$r_i = (h_{i1} : p_{i1}) \vee \ldots \vee (h_{in} : p_{in}) \leftarrow b_{i1}, \ldots, b_{im},$$

where $\pi = \{p_{ij}\}_{i,j}$ are the unknown parameters to be estimated from a set of training sequences D. A standard approach is to find max-likelihood parameters $\pi^* = \arg\max_\pi P(D \mid \pi)$. To determine a model parameter p_{ij}, we essentially need to know the number of times head element h_{ij} has been selected in an application of the rule r_i in the training data, which will be denoted by κ_{ij}. However, the quantity κ_{ij} is not directly observable. To see why this is so, first consider a single transition $I_t \rightarrow I_{t+1}$ 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 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 every transition, and κ_{ij} the sufficient statistics. To keep the notation uncluttered, we present the expectation step $\mathbb{E}[\kappa_{ij} \mid \pi, D]$ for a single transition $\tau = I_t \rightarrow I_{t+1}$; contributions from different transitions and different training sequences simply sum up. Let $\Gamma = \{\sigma \mid I_t \xrightarrow{\sigma} I_{t+1}\}$ denote the set of selections yielding τ. The expectation is

$$\begin{aligned}
\mathbb{E}[\kappa_{ij} \mid \pi, \tau] &= \sum_\sigma P(\delta_{ij} \mid \sigma, \pi, \tau) \\
&= \sum_\sigma P(\delta_{ij} \mid \sigma) P(\sigma \mid \pi, \tau) \\
&= \sum_{\sigma \in \Gamma} P(\delta_{ij} \mid \sigma) \frac{P(\sigma \mid \pi)}{\sum_{\sigma' \in \Gamma} P(\sigma' \mid \pi)} \qquad (4)
\end{aligned}$$

where δ_{ij} is an indicator variable representing that head h_{ij} was selected in rule r_i. Note that $P(\delta_{ij} \mid \sigma)$ is simply 1 if the head is selected in σ and 0 otherwise, and $P(\sigma \mid \pi)$ is defined by Equation 1. Given the expectation, the maximization step is

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

The key algorithmic challenge is to compute the expectation given by Equation 4 efficiently. As outlined above, the set Γ of selections yielding the observed transitions can be compactly represented as the set of paths from the root to the 1-terminal in a (possibly zero-suppressed) decision diagram.

By analogy to the inference problem, the summation given by Equation 4 can be performed in linear time given the BDD (ZDD) structure. This is realized by a dynamic programming algorithm similar to the forward-backward algorithm in hidden Markov models [3] that sweeps through the BDD structure twice to accumulate the sufficient statistics κ_{ij}. Details of the algorithm are straightforward but somewhat involved, and omitted for lack of space. Note that the presented Expectation-Maximization algorithm, by taking the special structure of our model into account, is significantly more efficient than general-purpose parameter learning techniques employed in CP-logic.

3.3 Prediction

Assume we are given a (partial) observation sequence $I_0, ..., I_t$, a CPT-theory $\mathcal{T}$, and a property of interest F (represented as a first-order formula), and would like to compute $P(I_{t+d} \models_B F \mid I_0, ..., I_t, \mathcal{T})$. 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. Note that the representation as a first-order formula allows one to express richer world conditions than queries on (sets of) atoms, as they are typically supported in statistical relational learning systems. In CPT-L,

$$P(I_{t+d} \models_B F \mid I_0, ..., I_t, \mathcal{T}) = P(I_{t+d} \models_B F \mid I_t, \mathcal{T})$$

as the world model is Markov. Powerful statistical relational learning systems are in principle able to compute this quantity 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 $I_{t+1}, ..., I_{t+d-1}$. In contrast, inference in CPT-theories draws its efficiency from the full observability assumption.

As an alternative approach, we propose a straightforward sample-based approximation to $P(I_{t+d} \models_B F \mid I_t, \mathcal{T})$. Given I_t, independent samples can be obtained from the conditional distribution $P(I_{t+1}, ..., I_{t+d} \mid I_t, \mathcal{T})$ by simply sampling according to $\mathcal{T}$ from the initial state I_t. Ignoring $I_{t+1}, ..., I_{t+d-1}$ and checking F in I_{t+d} yields independent samples of the boolean event $I_{t+d} \models_B F$ from the distribution $P(I_{t+d} \models_B F \mid I_t, \mathcal{T})$. The proportion of positive samples of this variable will thus quickly approach the true probability $P(I_{t+d} \models_B F \mid I_t, \mathcal{T})$.

4 Experimental Evaluation

The proposed CPT-L model has been evaluated in two different domains. First, we discuss experiments in a stochastic version of the well-known *blocks world* domain, an artificial domain that allows to perform controlled and systematic experiments e.g. with regard to the scaling behavior of the proposed algorithms. Second, the model is evaluated on real-world data collected from a live server of a massively multi-player online strategy game. Experiments in these two domains will be presented in turn.

4.1 Experiments in a Stochastic Blocks World Domain

As an artificial test bed 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. Application scenarios involving agents that act and learn in an environment are one of the main motivations for CPT-L. In such scenarios world-transition dynamics typically stem from *actions* carried out by the agents according to some *policy*. In the blocks-world domain discussed in this section, we assume that the policy of the agent is known and the task is to probabilistically model transition dynamics given the policy. It is straightforward to represent such conditional world models in CPT-theories by including the policy as part of the background knowledge.

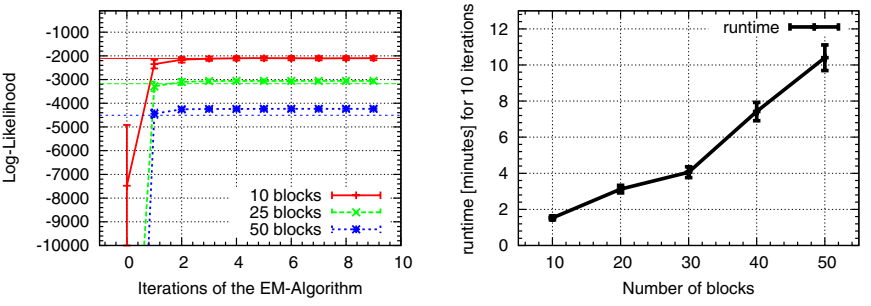

Fig. 2. Left graph: per-sequence log-likelihood on the training data as a function of the EM iteration. Right graph: Running time of EM as a function of the number of blocks in the world model.

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 $N \in \{1, ..., 4\}$. A predicate $free(B) \leftarrow not(on(A, B))$ 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 derived from the policy 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

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

they describe that stacks can start to jiggle if blocks are added to or taken from a stack. Furthermore, we consider 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 the stack never collapsed.

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, which 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 2, 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 2, 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 blocks[2]. This is in contrast to other, more expressive modeling techniques which typically scale badly to domains with many objects. The theory learned (Figure 2) 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 a Massively Multi-player Online Game

As an example for a massively multi-player online game, we consider *Travian*[3], a commercial, large-scale strategy game with a player community of about 3.000.000 players worldwide. In Travian, players are spread over several independent *game worlds*, with approximately 20.000–30.000 players interacting in a single world. Travian game play follows a classical strategy game setup. A game world consists of 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 game play elements, there are high-level aspects of game play 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 3 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

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

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

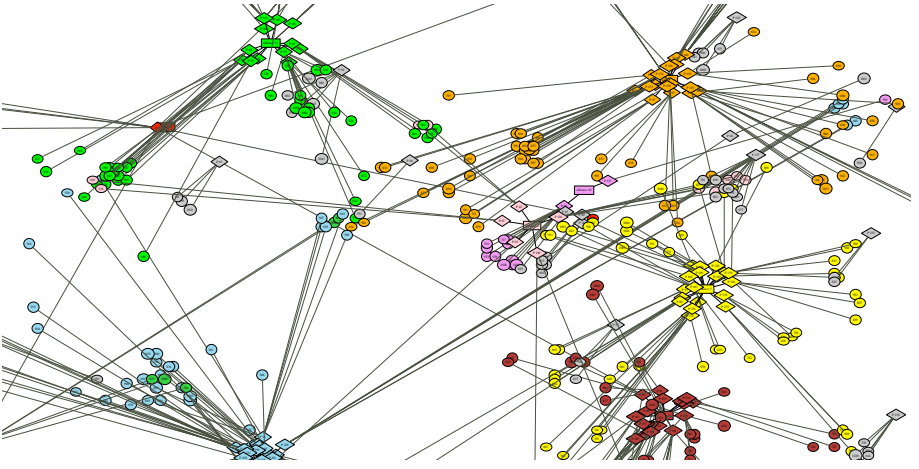

Fig. 3. High-level view of a (partial) game world in Travian. Circular nodes indicate cities, shown in their true positions on the game's grid-map. Diamond-shaped nodes indicate players, and are connected to all cities currently owned by the player. Rectangular nodes indicate alliances, and are connected to all players currently members of the alliance. Moreover, players and cities are color-coded according to their alliance affiliation.

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 Figure 4. Here, three players from the pink alliance launch a concerted attack against territory currently held by the green and orange 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. From all available data, we 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.

World Model. 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

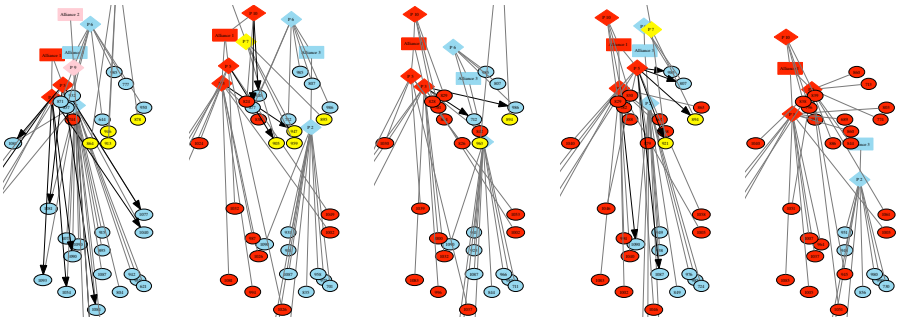

Fig. 4. Travian game dynamics visualized as changes in the game graph (for $t = 1, 2, 3, 4, 5$). Bold arrows indicate conquest attacks by a player on a particular city.

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. The final state descriptions (game graphs) on average contain approximately 50 objects (nodes) at every step in time, and relations between them. Sequences consists of between 29 and 31 such state descriptions.

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 manually defined a number of simple rules capturing such typical game patterns. As an example, consider the rules

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

The first rule encodes that a player is likely to conquest a city of a player he 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

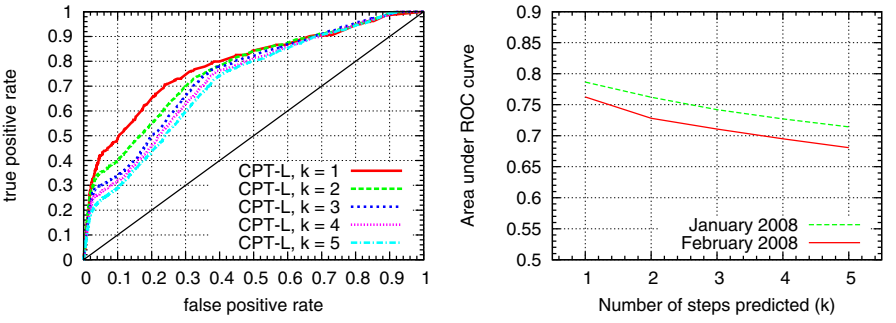

Fig. 5. Left figure: ROC curve for predicting that a city C will be conquered by a player P within the next k time-steps, for $k \in \{1, 2, 3, 4, 5\}$. The model was trained on 10 sequences of local game state descriptions from December 2007, and tested on 10 sequences from January 2008. Right figure: 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.

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.

Results in the Travian Domain. We consider the task of predicting the "conquest" action $conq(P, C)$ based on a learned generative model of world dynamics. The collected sequences of (local) 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 $I_0, ..., I_T$. For every $t_0 \in \{0, ..., T-1\}$, 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, $P(I_{t_0+1} \models conq(p, c) \mid \mathcal{T}, I_0, ..., I_{t_0})$. This probability is obtained from the sampling-based prediction algorithm described in Section 3. 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

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

This quantity is also easily obtained from the prediction algorithm for CPT-L. Figure 5, 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 5, 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 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.

Parameter learning for the CPT-L theory $\mathcal{T}$ on the training set takes approximately 30 minutes, and the model needed 5 iterations of EM to converge. Predicting the probability of $conq(p, c)$ for all player/city pairs and the next k time-steps starting from a particular world state takes approximately 1 minute.

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 [14], CP-logic [9], Probabilistic Relational Models [15] or Bayesian Logic Programs [16] 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 [17], 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). In contrast, such scenarios are naturally modeled in CP-logic and thus CPT-L. Another approach designed to model sequences of relational state descriptions are relational simple-transition models [18]. In contrast to CPT-L, they focus on domains where the process generating the data is hidden, and inferring these hidden states from observations. This is a harder setting than the fully observable setting discussed in this paper, and typically only approximate inference is possible [18].

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. This is essential for many real-life applications. The main direction for future work is to further evaluate the trade-off between representational power and scaling behavior in challenging real-world domains. Furthermore, we want to explore how the model can be extended, for instance to account for hidden data, without sacrificing efficiency.

Acknowledgments. The authors would like to thank the anonymous reviewers for helpful comments. This work was supported by FWO-Vlaanderen, and the GOA/08/008 project "Probabilistic Logic Learning".

References

1. Pollack, M.E.: Intelligent technology for an aging population: The use of AI to assist elders with cognitive impairment. AI Magazine 26(2), 9–24 (2005)
2. Laird, J.E., van Lent, M.: 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 (2000)
3. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
4. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., Chichester (1994)
5. Ghahramani, Z.: Learning dynamic bayesian networks. Adaptive Processing of Sequences and Data Structures. In: International Summer School on Neural Networks, pp. 168–197 (1997)
6. Fikes, R.E., Nilsson, N.J.: STRIPS: a new approach to the application of theorem proving to problem solving. In: Computation & intelligence: collected readings, Menlo Park, CA, USA, pp. 429–446. American Association for Artificial Intelligence (1995)
7. Getoor, L., Taskar, B. (eds.): Statistical Relational Learning. MIT Press, Cambridge (2007)
8. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.): Probabilistic Inductive Logic Programming. LNCS (LNAI), vol. 4911. Springer, Heidelberg (2008)
9. Vennekens, J., Denecker, M., Bruynooghe, M.: Representing causal information about a probabilistic process. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 452–464. Springer, Heidelberg (2006)
10. Bratko, I.: Prolog Programming for Artificial Intelligence, 2nd edn. Addison-Wesley, Reading (1990)
11. De Raedt, L., Kimmig, A., Toivonen, H.: Problog: A probabilistic Prolog and its application in link discovery. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 2462–2467 (2007)
12. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers 35(8), 677–691 (1986)
13. Minato, S.: Zero-suppressed BDDs for set manipulation in combinatorial problems. In: DAC 1993: Proceedings of the 30th international conference on Design automation, pp. 272–277. ACM, New York (1993)
14. Richardson, M., Domingos, P.: Markov Logic Networks. Machine Learning 62, 107–136 (2006)
15. Getoor, L., Friedman, N., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: Relational Data Mining, pp. 307–335. Springer, Heidelberg (2001)
16. Kersting, K., De Raedt, L.: Bayesian Logic Programming: Theory and tool. In: Getoor, L., Taskar, B. (eds.) An introduction to statistical relational learning. MIT Press, Cambridge (2007)
17. Zettlemoyer, L.S., Pasula, H., Kaelbling, L.P.: Learning planning rules in noisy stochastic worlds. In: Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005), pp. 911–918 (2005)
18. Fern, A.: A simple-transition model for relational sequences. In: Kaelbling, L., Saffiotti, A. (eds.) IJCAI, pp. 696–701. Professional Book Center (2005)