A Penalty-Logic Simple-Transition Model for Structured Sequences

Alan Fern AFERN@EECS.ORST.EDU
School of Electrical Engineering and Computer Science, Oregon State University

Abstract

We study the problem of learning to infer hidden state sequences of processes whose states and observations are propositionally or relationally factored. Unfortunately, standard exact inference techniques such as Viterbi and graphical model inference exhibit exponential complexity for these processes. The main motivation behind our work is to identify a restricted space of models, which facilitate efficient inference, yet are expressive enough to remain useful in many applications. In particular, we present the penalty-logic simple-transition model, which utilizes a very simple-transition structure where the transition cost between any two states is constant. While not appropriate for all complex processes, we argue that it is often rich enough in many applications of interest, and when it is applicable there can be inference and learning advantages compared to more general models. In particular, we show that sequential inference for this model, that is, finding a minimum-cost state sequence, efficiently reduces to a single-state minimization (SSM) problem. We then show how to define atemporal cost models in terms of penalty logic, or weighted logical constraints, and how to use this representation for practically efficient SSM computation. We present a method for learning the weights of our model from labeled training data based on Perceptron updates. Finally, we give experiments in both propositional and relational video-interpretation domains showing advantages compared to more general models.

1. Introduction

We consider hidden-state inference from the observations of processes with propositionally and relationally factored states and observations. That is, processes where states and observations can be described in term of propositions, objects, and relations among them. Unfortunately, standard exact inference techniques such as Viterbi (Forney, 1973) and variable elimination for graphical models (Dechter, 1999) exhibit exponential complexity for these processes. The main motivation behind our work is to identify a restricted space of models, which facilitate efficient inference, yet are expressive enough to remain useful in many applications. In particular, we note that in many domains it is possible to specify and/or learn rich logical constraints on states and observations that are rarely violated, though not perfectly consistent with reality. Our model is motivated by the desire to provide a framework for utilizing such constraints for robust sequential inference.

We introduce the *penalty-logic simple-transition model* which is parameterized by a set of weighted logical constraints and a single state-transition cost.[1] Under this model, the cost of a state sequence given an observation sequence is the weight of unsatisfied constraints plus a transition cost for each state change. Note that this model uses a very simple-transition structure that assigns a uniform cost for transitioning from any state to any other state.

1. This model was originally introduced in the conference version of this work (Fern, 2005).

While this type of transition model is not always appropriate, as argued later in this paper, for many processes of interest it is adequate. Intuitively, for many sequential processes, states persist for many time steps, and it is possible to robustly infer the state based on just the observations it generates without considering neighboring states. In such cases, our model effectively segments the observation sequence and utilizes the weighted logical constraints to reliably infer the state corresponding to each segment. In this paper, we study both learning and inference for this model.

We first study sequential inference for general simple-transition models (STMs). We show that this problem can be efficiently reduced to what we call the single-state minimization (SSM) problem, which involves inferring the least cost single state for an observation sequence. This result indicates that we can obtain efficient sequential inference assuming efficient SSM. We also show that STMs are somewhat distinguished with respect to efficient SSM. In particular, we show that for even a small extension to the STM model, no such efficient reduction is possible unless P=NP.

The next contribution of the paper is to show how to represent the atemporal part of our model in terms of penalty-logic theories, which are simply sets of weighted logical constraints. With such a representation, we show an approach to practically efficient SSM using a combination of logical reasoning and bounded search. The result is a robust way of utilizing nearly-sound logical constraints for sequential inference. We note that when a significant number of constraints are not nearly-sound, i.e. highly probabilistic, our methods are still applicable, but performance may suffer in terms of efficiency and/or accuracy.

We also describe a simple approach to learning the parameters of a penalty-logic-based STM from labeled training data. Given a set of logical constraints, either provided or learned by other mechanisms, we show how to instantiate a sign-constrained variant of the structured Perceptron algorithm (Collins, 2002) to jointly learn both the constraint weights and transition cost of the model. We show that this algorithm converges in a finite amount of time given usual margin assumptions.

We evaluate the learning and inference method in two real domains. The first, is a propositional domain, studied previously by Torralba, Murphy, Freeman, and Rubin (2003), which involves inferring the sequence of physical locations based on observations from a head-mounted web-cam. The second domain is relational and involves inferring relational force-dynamic states from video sequences. The results show that our model is effective and has advantages compared to a number of more general models.

In what follows, in Section 2, we first describe our problem setup and give an example of the type of sequential inference problem we are interested in. In Section 3 we introduce the simple-transition model. Section 4 then discusses inference for this model, showing the efficient reduction to SSM and pruning mechanisms. In Section 5, we present a hardness result for inference in non-simple models, showing that STMs are somewhat distinguished with respect to reducibility to SSM. In Section 6 we introduce our propositional penalty-logic representation for atemporal cost models and show how to perform practically efficient SSM. Section 7 extends to relational representations via the introduction of penalty-logic schemas. Section 8, describes how to apply the structured Perceptron algorithm to the problem of learning the parameters of a simple-transition model. Section 9 presents experimental results in two domains. Finally, Section 10 reviews related work.

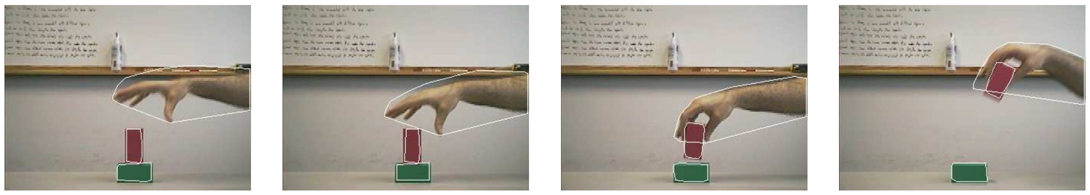


Figure 1: Key frames in a video segment showing a hand picking up a red block from a green block that is laying on the table. The object tracker's output is shown by the polygons. The video segment has two distinct force-dynamic states given by: {Grounded(hand), Grounded(green), Contacts(green, red)} (frames 1 and 3) and {Grounded(hand), Grounded(green), Attached(hand, red)} (frames 14 and 20). The transition occurs between frames 3 and 14. See Example 2 regarding the predicates Grounded, Contacts, and Attached.

2. Problem Setup

For simplicity, we describe our problem setup and approach for propositional (rather than relational) processes, where in the spirit of dynamic Bayesian networks (DBNs) (Murphy, 2002), states are described by a fixed set of variables. In Section 7, we extend to the relational setting.

We are interested in the problem of sequential inference where we are presented with observation sequences and asks to label them by corresponding state sequences. Here observation sequences (o-sequences) and state sequences (s-sequences) are simply finite sequences over elements of the *observation space* $\mathcal{X}$ and *state space* $\mathcal{Y}$ respectively. Throughout we assume that for any o-sequence the target state sequence has the same length and thus the problem can be viewed as predicting a state label for each element of an o-sequence. By convention, for a sequence $Q = (q_1, \ldots, q_T)$, we let $Q_{i:j} = (q_i, \ldots, q_j)$ for $i \leq j$. We will use uppercase for sequences and lowercase for single states and observations. We say that a sequential inference problem is *propositional* when its states are represented using a set of n *state variables* over the finite domain D_s, yielding $\mathcal{Y} = (D_s)^n$. The value of the i'th variable in state s is denoted by $s^{(i)}$. We will not need to assume a representation for observations until Section 7.

In this paper, we are interested in the case where we are not provided with an explicit description of the process that generates observation and corresponding state sequences. Rather, we assume that we are given a training set of corresponding s-sequence/o-sequences pairs drawn from the process of interest. We take a machine learning approach to solving this problem, where we learn a model of the process from the training set and then use that model to infer s-sequences for newly observed o-sequences. The focus of this paper is on describing a particular model for propositional and relational processes along with the associated inference and learning algorithms.

Example 1. *The process in our video-interpretation domain corresponds to a hand playing with blocks. Figure 1 shows key frames where a hand picks up a red block from a green block that is laying on the table. The goal is to observe the video and then infer* *the underlying force-dynamic states, where the force-dynamic state describes the support relations among objects. States are represented as sets of force-dynamic facts, such as* ATTACHED(HAND, RED)*. Observations are represented as sets of facts quantifying basic geometry relations and object properties in the scene, such as* DISTANCE(GREEN, RED) = 3*, that are derived from an object tracker's noisy output. For a fixed set of objects, the process can be represented propositionally with a variable for each possible fact. However, our system must handle any number of objects, requiring a relational process representation described in Section 7.*

3. The Simple-Transition Cost Model

Our framework utilizes an additive conditional cost model $C(S_{1:T}|O_{1:T})$ to represent the cost of labeling o-sequence $O_{1:T}$ by s-sequence $S_{1:T}$

$$C(S_{1:T}|O_{1:T}) = \sum_{1\leq i\leq T} C_a(s_i|o_i) + \sum_{1<i\leq T} C_t(s_i, s_{i-1}) \tag{1}$$

where $C_a : \mathcal{Y} \times \mathcal{X} \to R$ is an *atemporal-cost function*, $C_t : \mathcal{Y} \times \mathcal{Y} \to R$ is a *transition-cost function*, and $\mathcal{Y}$ and $\mathcal{X}$ are the state and observations spaces of the process under consideration. Sequential inference involves computing $\arg\min_{S_{1:T}} C(S_{1:T}|O_{1:T})$ for a given $O_{1:T}$.

This model has a familiar Markovian form, corresponding to the condition version of the cost model implicit in hidden Markov models and two-time-slice DBNs (Murphy, 2002). Intuitively, $C_a(s_i|o_i)$ represents the local cost of labeling the i'th sequence position as s_i given that we observed o_i, and $C_t(s_i, s_{i-1})$ represents the cost of transitioning from state s_{i-1} to s_i. For simplicity, we only allow C_a to depend on a single local observation. However, as for conditional random fields (Lafferty, McCallum, & Pereira, 2001), all of our discussions generalize to the case where the local cost of s_i can depend on the entire observation sequence rather than just o_i.

In this work, we focus on models with a particularly simple-transition-cost function. We say that $C(S|O)$ is a *simple-transition model* if $C_t(s_i, s_{i-1}) = K \cdot \delta(s_i \neq s_{i-1})$, where $\delta(p)$ is 1 if p is true and 0 otherwise, and K is a positive constant. Thus, a simple-transition model can be parameterized by the pair $\langle K, C_a \rangle$. This model charges a constant cost K for each state transition, regardless of the states involved.

Clearly, it is generally the case that some state transitions are more likely than others. If this likelihood information is critical for accurate inference, then a simple-transition model will not suffice on its own. However, there are interesting classes of processes where accurate inference is possible without exploiting non-simple-transition structure. For example, in processes such as our video-interpretation domain, states tend to persist for many observations, and a state can be reliably inferred by integrating only the observations that it generates, without considering neighboring states. In such cases, sequential inference can roughly be viewed as inferring the approximate state transition points and then integrating the observations between transitions to infer states. Intuitively this type of inference can be captured by the simple-transition model—the value of K can be thought of as indicating the degree to which we expect states to persist. For such processes it is important to study simple but sufficient models, as there can be considerable computational advantages, compared to using more general transition models such as DBNs. In addition, when STMs are well suited to a domain, there can be advantages with respect to generalization, compared to more complex models with many more parameters.

Note that the value of K plays an important role in the resulting state inference. For very small values of K the infered state sequences will not reward temporal continuity and hence will be more susceptible to noisy observations. At the other extreme, for very large values of K, the model will care more about temporal continuity rather than the local information provided by observations. STMs allow for a spectrum of inference behavior between these extremes and thus provides a parameterized structure for realizing the above segmentation-style of inference. Our learning approach introduced later in the paper automatically optimizes the value of K so as to maximize the accuracy of inference on the training set, relieving the user from this difficult task.

At first blush, STMs might appear to be quite implausible for problems that involve many state variables, since if a single variable changes there is no additional cost from the transition model for any of the the other variables to change their values arbitrarily. However, it is important to note that the additional cost for such arbitrary changes will typically be charged by the observation model since the changes will not agree with the observed evidence as integrated over the segments between transitions.

Finally, we note that in addition to our blocks-world video-interpretation domain, we are considering a number of other domains for which the simple-transition model appears well suited. Some of these include:

- **Sports Video Interpretation.** Professional and college American football teams spend a great deal of effort attaching semantic tags to football video in order to facilitate fast semantic indexing by coaches during game planning. Automating the interpretation process would be a valuable tool. Many properties of interest in sports video, such as American football, evolve at a much slower time scale than frame rate. For example, a defender will typically cover a particular offensive receiver for many video frames, or in basketball, a player dribbles a ball for many video frames. While such properties may not be reliably inferable from a single observation, such properties produce distinctive sequences of frames, such that taken together can aid reliable interpretation.

- **Personal Location Tracking.** The work in (Torralba et al., 2003) considers using hidden Markov models to track the high-level location of a person wearing a head mounted camera—e.g. inferring the room number or street name. In this application, each location generates many video frames that are highly indicative of the particular location. Our experiments with this data later in the paper verify that the simple-transition model is suitable and generalizes better than a more general model.

- **Motion Capture Interpretation.** Another domain is the interpretation of multi-object motion capture data. In particular, for interpreting the scenes in an assembly tutoring system. Here motion-capture sensors are attached to the human subjects, various assembly objects, and tools. In order to interpret the scenes it is critical to infer the slowly changing force-dynamic properties based on the noisy marker position data.

Here each force-dynamic configuration gives rise to a large number of observations that are highly indicative of the particular configuration.

- **Task Prediction for Intelligent User Interfaces.** The TaskTracer system (Dragunov, Dietterich, Johnsrude, McLaughlin, Li, & Herlocker, 2005) is an intelligent user interface that is structured around the idea of user tasks. An important component of such a system is the ability to infer the current task of the user in order to provide task specific assistance. In this application, the observations correspond to all of the user interface events captured by the system, and for any particular task there are typically a large number of observations that are indicative of that task. Preliminary work in (Shen, Li, & Dietterich, 2007) verifies that STMs are well suited to this task.

4. Inference for Simple-Transition Models

In this section, we consider sequential inference for the simple-transition model, i.e. computing $\arg\min_{S_{1:T}} C(S_{1:T}|O_{1:T})$. Here we treat C_a as a generic function, although its representation is critical for efficient inference and will be introduced in Section 6, where we describe how to represent atemporal cost models using penalty logic.

Since the simple-transition model can be viewed as a conditional hidden Markov model (HMM), we can apply the standard Viterbi algorithm (Forney, 1973) to compute the minimum-cost state sequence $S_{1:T}$ in $O(T \cdot |\mathcal{Y}|^2)$ time. In fact, it is possible to improve this computation by leveraging the simple-transition structure in order to obtain an $O(T \cdot 2 \cdot |\mathcal{Y}|)$ time algorithm.[2] However, as is well noted for DBNs, these algorithms are impractical for non-trivial propositional processes as the state-space scales exponentially in the number of state variables n, i.e. $|\mathcal{Y}| = |D_s|^n$. Likewise, general-purpose graphical-model inference techniques such as variable elimination and junction-tree algorithms exhibit exponential behavior in n for the simple-transition model. This is due to the $\delta(s_i, s_{i-1})$ terms in the simple-transition model, which causes the induced tree width (Dechter, 1999) of the corresponding graphical structure to be linear in n, indicating that standard graphical techniques will be exponential time in n. This exponential behavior is a result of ignoring the special structure of the simple-transition model. For the relational processes we are interested in, the effective value of n will typically scale at least quadratically with the number of objects, making these techniques inapplicable.

4.1 Reduction to Single-State Minimization

For a given STM C and o-sequence $O_{1:T}$ we denote the minimum-cost over all s-sequences by $C^*(O_{1:T}) = \min_{S_{1:T}} C(S_{1:T}|O_{1:T})$ and let the *witness set* $C^*_w(O_{1:T})$ be the set of optimal s-sequences that achieve cost $C^*(O_{1:T})$.

Our approach to sequential inference for STMs, i.e. computing $C^*(O_{1:T})$, is by reduction to a problem we will call *single-state minimization (SSM)*.

2. For time step i, the Viterbi algorithm considers the possibility of transitioning from each possible state at time $i-1$ to each possible state at time i, which is quadratic in the number of states. For the simple-transition model, at time i we only need to consider the possibility of being in the same state at time $i-1$ or not, which is linear in the number of states. This observation also follows by using distance transforms for Viterbi maximization as in (Felzenszwalb, Huttenlocher, & Kleinberg, 2003).

Definition 1 (Single-State Minimization). *Given an atemporal cost function C_a and an o-sequence $O_{1:T}$, the SSM problem is to compute the SSM cost function $\sigma(O_{1:T}|C_a) = \min_s \sum_{1\leq i\leq T} C_a(s|o_i)$ and the corresponding SSM witness function $\sigma_w(O_{1:T}|C_a) = s$ where state s is a state achieving $\sigma(O_{1:T}|C_a)$ (i.e. $\sigma(O_{1:T}) = \sum_{1\leq i\leq T} C_a(s|o_i)$). When C_a is clear from context we will denote the values of these functions by just $\sigma(O_{1:T})$ and $\sigma_w(O_{1:T})$.*

Solving SSM gives the minimum cost of labeling an o-sequence $O_{1:T}$ by a single state—i.e. the minimum cost label with no state transitions. An important property of STMs is that $C^*(O_{1:T})$ can be expressed in terms of the SSM cost function σ. In what follows we say that an s-sequence $S_{1:T}$ *has a final transition at* j, for $j > 0$, if $s_j \neq s_{j+1}$ and all states after s_j are equal. We say that $S_{1:T}$ has a final transition at $j = 0$ if $S_{1:T}$ has no transitions.

Proposition 1. *Given a STM C and o-sequence $O_{1:T}$, for any $0 \leq j < T$ we have that*

$$C^*(O_{1:T}) \leq C^*(O_{1:j}) + K\cdot\delta(j > 0) + \sigma(O_{j+1:T})$$

with equality if and only if there is a member of $C^*_w(O_{1:T})$ with final transition at j.

The proof is straightforward and is in the Appendix A. This proposition tells us that if an optimal s-sequence has a final transition at j then $C^*(O_{1:T})$ is equal to the minimum cost achievable up to position j, plus a transition cost K (unless $j=0$), plus the SSM cost for the remaining suffix (no transitions occur after j). That is, the computation of $C^*(O_{1:T})$ decomposes into two completely decoupled minimization problems, one for the prefix up to j and another for the postfix after j. This critical decomposition is possible because STMs weigh all transition types equally.

Proposition 1 also tells us that for any $0 \leq j < T$, $C^*(O_{1:T})$ will not be greater than the expression $C^*(O_{1:j}) + K\cdot\delta(j > 0) + \sigma(O_{j+1:T})$. Since there must be some index j that is the final transition of an optimal s-sequence, we can compute $C^*(O_{1:T})$ by minimizing this expression over all j, which gives the following key recursion.

$$C^*(O_{1:T}) = \min_{0\leq j<T} [C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})] \tag{2}$$

Equation 2 yields an efficient reduction to SSM based on dynamic programming, which we call the *SSM-DP algorithm*. Simply compute $C^*(O_{1:t})$ in the order $t = 1, 2, \ldots, T$ for a total of T^2 SSM computations. It is straightforward to store information for extracting a minimum-cost s-sequence in $C^*_w(O_{1:T})$ (via calls to σ_w) and we will denote any such sequence by SSM-DP$(O_{1:T}, \sigma, K)$.

We can view Equation 2 as the recursion used for inference in segment models (Ostendorf, Digalakis, & Kimball, 1996), specialized for our simple-transition structure. However, like the Viterbi algorithm, segment-model inference typically involves enumerating the state space. In our case this corresponds to computing the SSM function σ by considering each possible state to find the one of minimum cost. Computing SSM in this way has a time complexity of at least $O(T \cdot |D_s|^n)$, and thus the SSM-DP algorithm would require $O(T^3 \cdot |D_s|^n)$ time, which is exponential in n, and not practical for our purposes.

At first blush it does not seem that our reduction to SSM has gained us anything, in fact the complexity is now cubic in T, rather than linear as for Viterbi. However, the utility of this reduction, for large n, becomes clear when SSM can be computed efficiently relative to

n. That is, efficient SSM computation implies efficient sequential inference for the simple-transition model. The complexity of SSM depends on the particular representation used for the atemporal cost function C_a. The basis of our approach is to use a representation grounded in weighted logical constraints, or penalty logic, in order to provide practically efficient SSM.

4.2 Pruning for SSM-DP

Though our reduction provides the potential for handling large state and observation spaces, the naive DP reduction requires $O(T^2)$ calls to SSM, which will be unacceptable for many applications. Here we describe a very effective pruning technique that we have empirically observed to reduce the number of SSM computations to $O(T)$.

The $O(T^2)$ complexity arises from the fact that when computing $C^*(O_{1:t})$, Equation 2 dictates that we consider all previous times $j < t$ as possible locations for the final state transition. Fortunately, in practice, it is possible to soundly eliminate most values of j from consideration as DP progresses. Instead of computing $C^*(O_{1:t})$ by considering each value of $j < t$ we maintain a list L_t of j values that have *not* yet been eliminated by pruning. When computing $C^*(O_{1:t})$ we only consider $j < t$ that are also in L_t. Intuitively, after computing $C^*(O_{1:t})$, if we find this cost is small compared to the optimal cost up to index j plus the SSM value from j to t, then we can eliminate j from consideration as a final transition point. This is because t could always be used instead of j as a final transition point and achieve as good or better cost.

More specifically, the sets L_t are updated as follows:

- $L_1 = \{0\}$
- $L_{t+1} = \{t\} \cup \{j \in L_t \mid C^*(O_{1:t}) > C^*(O_{1:j}) + \sigma(O_{j+1:t}) - K \cdot \delta(j = 0)\}$

That is, after completing DP iteration t we can eliminate any value $j < t$ from L_t that satisfies $C^*(O_{1:t}) \leq C^*(O_{1:j}) + \sigma(O_{j+1:t}) - K \cdot \delta(j = 0)$. The resulting algorithm is summarized in Figure 2. The soundness of this pruning rule is given by the following proposition which shows that if we remove an index from L_t, then there will be another index in L_{t+1} that is at least as good.

Proposition 2. *Given a STM C and an o-sequence $O_{1:T}$, let the sets L_t be computed as described above. For any $1 \leq t < T$ and any $0 \leq j < t$, if $j \in L_t$ and $j \notin L_{t+1}$ (i.e. j was removed on iteration t), then there is a $j' \in L_{t+1}$ such that for any $t' > t$ if $C^*_w(O_{1:t'})$ contains an s-sequence with final transition at j then it also contains an s-sequence with final transition at j'.*

The proof is in the Appendix. This proposition tells us that during SSM-DP, when computing $C^*(O_{1:t})$, it is safe to only consider minimizing over value of j that are in L_t. In our video-interpretation experiments, this pruning mechanism dramatically reduces the number of j values considered, typically to less than ten per DP step.

```
PRUNED-SSM-DP($O_{1:T}, \sigma, K$)
$L \leftarrow \{0\}$;
for  $t = 1, \ldots, T$
    for-each $j \in L$
        Compute and store $\sigma(O_{j+1:t})$
    Compute and store $C^*(O_{1:t}) = \min_{j \in L} [C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:t})]$
    $L \leftarrow L \cup \{t\} - \{j \in L : C^*(O_{1:t}) \leq C^*(O_{1:j}) + \sigma(O_{j+1:t}) - K \cdot \delta(j = 0)\}$
return $C^*(O_{1:T})$
```

Figure 2: The SSM-DP algorithm with the incorporation of sound pruning.

4.3 Sufficient SSM Approximation

In practice we do not need to compute the exact SSM function for all possible o-sequences. Rather we need only to compute a sufficient SSM approximation as we define below. Later we will translate this observation into practical computational savings.

We say that $O_{i:j}$ is a *critical o-sequence* of a sequential inference problem $\mathcal{P}$ if for some possible o-sequence $O_{1:T}$ there is an optimal target s-sequence $S_{1:T}$ in which $O_{i:j}$ is a maximal subsequence generated by a single state. Let σ be an SSM function corresponding to some STM, presumably one that accurately predicts s-sequences of the sequential inference problem $\mathcal{P}$. We say σ' is a *sufficient SSM approximation* to σ for $\mathcal{P}$ if both 1) for all o-sequences O, $\sigma(O) \leq \sigma'(O)$, and 2) $\sigma'(O') = \sigma(O')$ for any critical o-sequence O' of $\mathcal{P}$. It is straightforward to show that we can do as well at predicting s-sequences of $\mathcal{P}$ using σ' as we can using σ.

Proposition 3. *Let σ' be a sufficient SSM approximation to σ for sequential inference problem $\mathcal{P}$, and assume that SSM-DP breaks ties according to a fixed ordering over s-sequences. For any possible $O_{1:T}$ and corresponding target $S_{1:T}$ arising in $\mathcal{P}$, if SSM-DP($O_{1:T}, \sigma, K$) $= S_{1:T}$ then SSM-DP($O_{1:T}, \sigma', K$) $= S_{1:T}$.*

Proof. First we introduce the concept of "cost under an SSM function". Given an o-sequence $O_{1:T}$ and an s-sequence $S_{1:T}$ we denote by $\mathcal{O}(O_{1:T}, S_{1:T}) = \{O_1, O_2, \ldots, O_m\}$ the set of maximal o-sequences of $O_{1:T}$ for which there is no state transition in $S_{1:T}$. For a given K we define the *cost under SSM function* σ of $S_{1:T}$ given $O_{1:T}$ to be $(m-1) \cdot K + \sum_{1 \leq i \leq m} \sigma(O_i)$, which is equal to the cost assigned to $S_{1:T}$ given $O_{1:T}$ by an STM corresponding to σ. Note that for an arbitrary SSM function there need not be a corresponding STM—nevertheless, this notion of cost under an SSM function is well defined. SSM-DP returns an s-sequence that has minimum cost under its given SSM function.

Assume that $O_{1:T}$ is a possible o-sequence with target s-sequence $S_{1:T}$ with respect to $\mathcal{P}$. In this case we have that

$$\mathcal{O}(O_{1:T}, S_{1:T}) = \{O_1, O_2, \ldots, O_m\}$$

is the set of critical o-sequences of $O_{1:T}$. If SSM-DP($O_{1:T}, \sigma, K$) $= S_{1:T}$ then we know that there is no other s-sequence with a better cost under σ. Furthermore since σ' is a sufficient approximation we know that the cost of $S_{1:T}$ under σ' is equal to the cost under σ, and we also know that no s-sequence will have a lower cost under σ' than under σ. Thus, $S_{1:T}$ will be a minimum cost s-sequence under σ' and since ties are broken according to a fixed ordering we have that SSM-DP$(O_{1:T}, \sigma', K) = S_{1:T}$. $\square$

At this point it is unclear how one might compute a sufficient SSM approximation without already knowing the true s-sequences. In Section 6, we describe how this can be done for our constraint-based representation, and how it leads to computational benefits. Intuitively, our computation of $\sigma'(O)$ will be based on searching for a minimum-cost state for O, which will sometimes yield sub-optimal solutions with respect to $\sigma(O)$, in which case, we automatically satisfy the first condition for sufficient SSM. Furthermore, the search needs to only find optimal solutions for long enough state sequences to satisfy the second condition. For these sequences we will argue that often very little search is needed to find the optimal result, thus allowing for efficient and sufficient SSM computation.

5. Hardness for Non-Simple Models

Given our focus on SSM it is natural to consider efficient SSM reductions for non-simple models. Intuitively, if a model distinguishes between different transition types, we may need to consider states other than just SSM solutions (like Viterbi but unlike SSM-DP), possibly resulting in exponential behavior in n even given efficient SSM. Below we show that an efficient reduction is unlikely for a modest extension to STMs, giving a boundary between efficient and inefficient models under efficient SSM.

We extend STMs by allowing the model to assign higher costs to transitions where more state variables change, unlike STMs which can only detect whether some change occurred. A cost model C, with atemporal component C_a, is a *counting-transition model* if $C_t(s_i, s_{i-1}) = K \cdot \sum_{1 \leq j \leq n} \delta(s_i^{(j)} \neq s_{i-1}^{(j)})$, i.e. transition cost is linear in the count of propositions that change. We say C *allows efficient SSM* if the SSM function for C_a is computable in time polynomial in its input o-sequence size and number of state variables.

Theorem 1. *Given a counting-transition model C that allows for efficient SSM, an observation sequence $O_{1:T}$, and a cost bound τ, the problem of deciding if* $\min_{S_{1:T}} C(S_{1:T}|O_{1:T}) < \tau$ *is NP-complete.*

This result shows that STMs are somewhat distinguished with respect to their ability to leverage efficient SSM computation. In the remainder of this section, we present the proof of this result, which relies on a previous hardness results for 2-d grid Potts model inference. A *Potts model* is a tuple $\langle V, E, D, C_l, C_p \rangle$ where V and $E \subseteq V \times V$ are the vertex and edge sets of a finite graph, D is a finite domain of vertex labels, $C_l : V \times D \to \mathcal{R}$ is a *label cost function*, and $C_p : E \to \mathcal{R}$ is a *pairwise cost function*. We interpret $C_l[v, d]$ as the cost of labeling vertex v with label d, and $C_p[v_1, v_2]$ as the cost of assigning vertices v_1 and v_2 different labels. A vertex labeling $L : V \to D$ assigns each vertex a label from D, and the cost of a labeling L is $C(L) = \sum_{v \in V} C_l[v, L(v)] + \sum_{(v_1,v_2) \in E} C_p[(v_1, v_2)] \cdot \delta(v_1 \neq v_2)$. That is, the cost is the total labeling cost over individual vertices plus the total cost incurred for assigning neighboring vertices different values. Throughout this section we will assume that the cost of a vertex assignment can be evaluated efficiently. We say that a Potts model is a

2-d grid Potts model if the graph given by V and E corresponds to a 2-d rectangular grid. That is, vertices are arranged in a rectangular grid and there are edges between vertical and horizontal neighbors in the grid. Figure 4a in Appendix A shows a graphical depiction of a 2-d grid Potts model. Potts models can be viewed as a restricted class of discrete Markov random fields and 2-d models have been commonly used for image processing and computer vision, where vertices correspond to pixels.

The following hardness result, credited to Jon Kleinberg, shows that finding the minimum-cost labeling for Potts models is computationally hard, even for 2-d grids.

Theorem 2. *(Veksler, 1999) Given as input a 2-d grid Potts model and a threshold τ, the problem of deciding whether there is a vertex labeling with cost less than τ is NP-complete.*

The first step in our proof of Theorem 1 is to give a non-trivial extension of this hardness result to a more restricted class of 2-d grids where horizontal pairwise edge costs are all equal. We say that a 2-d grid Potts model is *h-constant* if $C_p[(v_1, v_2)] = K$ for all horizontally neighboring vertices v_1 and v_2 in the grid, for some constant K. For example, the model depicted in Figure 7 in Appendix A is an h-constant 2-d grid. The proof given for Theorem 2 does not extend to h-constant models as it relies on constructing models that are not h-constant. To the best of our knowledge there are no published results on the complexity of this restricted model class and our personal communication with Kleinberg and the authors of (Veksler, 1999) suggests that the problem was open before our work. Interestingly, it also appears the sub-class of our problem where all edge costs are equal (both vertical and horizontal) is still open. This is somewhat surprising given that in practice image processing and computer vision researchers often consider 2-d grid models where all edge costs are equal.

Lemma 1. *Given as input an h-constant 2-d grid Potts model and a threshold τ, the problem of deciding whether there is a vertex labeling with cost less than τ is NP-complete.*

The proof is in Appendix A. With this lemma in hand, we are now ready to prove Theorem 1 by reduction from h-constant 2-d grid models.

Proof. Membership in NP is again trivial. To show hardness consider an h-constant 2-d grid Potts model $P^* = \langle V, E, D, C_l, C_p \rangle$ where all horizontal edge costs (given by C_p) are equal to the constant K. Let T and n be the number of columns and rows respectively in P^*. We will denote the vertex in column t and row j as $v_t^{(j)}$.

Intuitively, our reduction will treat $v_t^{(j)}$ as the j'th propositional state variable at time t for some process, which we will denote by $s_t^{(j)}$. Note that if two neighboring vertices in corresponding columns $v_t^{(j)}$ and $v_{t+1}^{(j)}$ are labeled differently then there will be a cost of K, which corresponds exactly to the transition cost of the counting transition model (i.e. there is a cost of K for each state variable transition). Below we define an atemporal cost function for a counting transition model that corresponds exactly to P^* and at the same time allows for efficient SSM.

We assume that the observation space for which our counting transition model is defined has exactly T elements $\{o_1, \ldots, o_T\}$, and that the propositional state space is defined in terms of n state variables over the domain D. As required for counting transition models, we define the transition cost function as $C_t(s, r) = K \cdot \sum_{1 \leq j \leq n} \delta(s^{(j)} \neq r^{(j)})$. For each observation o_t, the atemporal cost function is defined as follows.

$$C_a(s|o_t) = \sum_{1 \leq j \leq n} C_l[v_t^{(j)}, s^{(j)}] + \sum_{1 < j \leq n} C_p[(v_t^{(j-1)}, v_t^{(j)})] \cdot \delta(s^{(j)} \neq s^{(j-1)}) \tag{3}$$

Intuitively, this equation calculates the atemporal cost for observation o_t by ignoring all of P^* except for column t and then calculating the cost of that column under the labeling given by s. For the o-sequence $O_{1:T} = (o_1, \ldots, o_T)$, it is straightforward to verify that the cost $C(S_{1:T}|O_{1:T})$ of a state sequence $S_{1:T}$ under this counting transition model is equal to the cost under P^* of the labeling $L(v_t^{(j)}) = s_t^{(i)}$. Since there is a one-to-one correspondence between length T state sequences $S_{1:T}$ and labelings of P^*, we see that there is a labeling for P^* with cost less than τ iff there is an $S_{1:T}$ such that $C(S_{1:T}|O_{1:T}) < \tau$. This completes the reduction from our decision problem for h-constant 2-d grid Potts models to the corresponding decision problem for counting transition models, showing NP-hardness.

It remains to show that the counting transition model we constructed allows for efficient SSM. The SSM function σ corresponding to the above atemporal cost function C_a can be expressed as follows.

$$\sigma(O_{q:r}) = \min_s \sum_{q \leq t \leq r} C_a(s|o_t) = \min_s \sum_{1 < j \leq n} \theta_j(s^{(j-1)}, s^{(j)}) \tag{4}$$

where,

$$\begin{aligned} \theta_j(s^{(j-1)}, s^{(j)}) &= \sum_{q \leq t \leq r} C_l[v_t^{(j)}, s^{(j)}] + C_p[(v_t^{(j-1)}, v_t^{(j)})] \cdot \delta(s^{(j)} \neq s^{(j-1)}) \\ &\quad + \delta(j = 2) \cdot \sum_{q \leq t \leq r} C_l[v_t^{(1)}, s^{(1)}] \end{aligned} \tag{5}$$

From Equation 4 we see that the SSM function σ can be expressed as minimizing a sum over the binary functions $\theta_j(s^{(j-1)}, s^{(j)})$. Note that the dependency structure on the state variables implied by these binary functions forms a length n linear chain. It is well known that we can use dynamic programming, in particular the Viterbi algorithm to solve the joint minimization problem for chains in time linear in the chain length and domain size $|D|$. Clearly all of the θ_i can be computed efficiently using Equation 5 and thus we can compute the SSM function efficiently. This shows that for an arbitrary h-constant 2-d grid Potts model we can construct an equivalent counting transition model that allows for efficient SSM. □

6. SSM Using Penalty-Logic Cost Models

We have seen that the efficiency of STM inference depends on the efficiency of SSM. In this section, we describe a logic-based representation for atemporal cost models that supports practically efficient SSM. We are motivated by the observation that in many domains, including our video-interpretation domain, there are nearly-sound logical constraints on states and between states and observations. In addition, it is often easy to automatically learn or provide such constraints. An important question then is how to best leverage those constraints for robust sequential inference. Any such method for doing this should be robust to the fact that the constraints will sometimes be violated, though typically they will be true. One such way for doing this is to attach weights to logical formulas and to treat the weights as costs of violating the constraints. This idea of using weighted logical constraints to represent cost models has been proposed previously by (Pinkas, 1991) under the name penalty logic. Below we describe our use of penalty logic for representing atemporal cost models and the associated SSM inference problem.

For simplicity, we assume that states have n binary state variables, i.e. $D_s = \{\mathbf{true}, \mathbf{false}\}$. We also assume a set of m binary *observation tests*, each one mapping observations to $\{\mathbf{true}, \mathbf{false}\}$. Non-binary extensions are straightforward. We will later extend to relational domains.

6.1 Propositional Horn Constraints

In this work, we focus on a subclass of logical constraints known as Horn constraints. While the ideas we present extend easily to arbitrary logical constraints, as we will see below, the use of Horn constraints will afford us improved tractability. A *propositional Horn constraint* ϕ is a logical implication ($body \rightarrow head$), where *body* is a conjunction of state variables and observation tests, and *head* is a state variable or **false**. Given an observation o and state s, we let $\phi[o]$ ($\phi[s]$) denote the result of substituting observation tests (state variables) in ϕ with the truth values under o (under s). If a constraint has no variables, then it is interpreted as the truth value of the variable-free expression. Thus, $\phi[o][s]$ is the truth value of ϕ under o and s, and we say that o and s *satisfy* ϕ iff $\phi[o][s]$ is **true**. A set of Horn constraints is *satisfiable* iff there exists a state and observation that jointly satisfy each constraint. Importantly, for Horn constraints, testing satisfiability and finding satisfying assignments is polynomial-time computable (Papadimitriou, 1995). This is in contrast to sets of arbitrary logical constraints for which test satisfiability is NP-hard.

6.2 Penalty-Logic Cost Functions

Penalty logic was developed by Pinkas (Pinkas, 1991, 1995) as a simple framework for dealing with logical inconsistency. Rather than view logical constraints as absolute truths about the world, penalty logic assigns a weight to each formula which indicates how much it "costs" to violate a constraint relative to others. This basic idea is related to an earlier proposal by Derthick (Derthick, 1990) with somewhat different semantics. A penalty-logic knowledge base is simply a set of weighted logical formulas. When the constraints are restricted to be Horn, we say that the set is a Horn penalty-logic knowledge base.

We will parameterize atemporal-cost functions using a Horn penalty-logic knowledge base $\Phi = \{\langle \phi_1, c_1 \rangle, \ldots, \langle \phi_v, c_v \rangle\}$, with Horn constraints ϕ_i and non-negative weights c_i representing the cost of violating ϕ_i. The non-negativity requirement will be important for inference as discussed below. The sum of costs in Φ is denoted by $\textsc{Cost}(\Phi)$ and we say Φ is satisfiable if its set of constraints is satisfiable. Atemporal cost is defined as $C_a(s|o, \Phi) = \sum_{\langle \phi, c \rangle \in \Phi} c \cdot \delta(\neg\phi[o][s])$, i.e. the total cost of unsatisfied constraints. In this work, we will assume that Φ contains "nearly sound" constraints, meaning that each constraint is usually satisfied by the state/observation pairs generated by our process. Our primarily non-theoretical goals do not require a formal notion of nearly sound (e.g. PAC).

Given Φ and o-sequence $O_{1:j}$, we define the *combined constraint set* as $\Gamma(O_{1:j}, \Phi) = \bigcup_{1 \leq i \leq j} \bigcup_{\langle \phi, c \rangle \in \Phi} \langle \phi[o_i], c \rangle$, which involves only state variables and captures all of the state constraints "implied" by Φ and $O_{1:j}$. The SSM function for C_a is now given by

$$\sigma(O_{1:j}|\Phi) = \min_s \sum_{1 \leq i \leq j} C_a(s|o_i, \Phi) = \min_s \sum_{\langle \phi, c \rangle \in \Gamma(O_{1:j}, \Phi)} c \cdot \delta(\neg \phi[s]) \tag{6}$$

which is equivalent to solving *maximum satisfiability (MAX-SAT)* (Jiang, Kautz, & Selman, 1995) for $\Gamma(O_{1:j}, \Phi)$, where MAX-SAT asks for an s such that the weight of satisfied constraints is maximum. While the number of SSM variables is fixed, the constraint set grows with sequence length. Fortunately, in practice we can significantly reduce this set by pruning and merging. Prune(Φ) contains members of Φ that do not have **false** in the constraint's body. Merge(Φ) combines logically equivalent members of Φ into one constraint by summing weights. MAX-SAT solutions are invariant under both operators. Thus, we solve SSM via MAX-SAT for the smaller constraint set $\Gamma^*(O_{1:j}, \Phi) = \textsc{Merge}(\textsc{Prune}(\Gamma(O_{1:j}, \Phi)))$.

Intuitively, for our constraints Φ to support accurate sequential inference for a process $\mathcal{P}$, we would like them to satisfy at least two properties. First, for any critical o-sequence $O_{1:j}$ of $\mathcal{P}$ (recall Section 4), the SSM solution should have a low cost and yield the single state s that generated $O_{1:j}$. Second, for o-sequences generated by more than one distinct consecutive state, we would like the SSM solution to have a higher cost so as to not be included in the SSM-DP solution.

It is often possible to find constraints that satisfy these properties, as is the case in our video-interpretation application. To see this, note that a critical o-sequence $O_{1:T}$ must be generated by a single state s. Intuitively, when our constraints are nearly sound, most all members of $\Gamma(O_{1:T}, \Phi)$ will be satisfied by s. This means that s will tend to have low cost and be an SSM solution for $O_{1:T}$ as desired. In the second case when $O_{1:T}$ is generated by more than one consecutive state, the constraints in $\Gamma(O_{1:T}, \Phi)$ will tend to conflict with one another, since neighboring states tend to have characteristics that are not entirely compatible. The result is that the SSM solution for such o-sequences will tend to violate more constraints and thus have higher costs.

6.3 A Dual MAX-SAT Approach

MAX-SAT is NP-hard even for Horn constraints (Jaumard & Simeone, 1988). Rather than use general-purpose approximate techniques, we give a MAX-SAT approach that leverages our setting of nearly-sound Horn constraints. Let $\pi = \Gamma^*(O_{1:j}, \Phi)$ and Π be the set containing all satisfiable subsets of π. An equivalent dual form of Equation 6 is $\sigma(O_{1:j}|\Phi) = \textsc{Cost}(\pi) - \max_{\pi' \in \Pi} \textsc{Cost}(\pi')$, which asks for a maximum-cost member of Π. This motivates the following MAX-SAT approach that searches through constraint subsets rather than variable assignments. Conduct a cost-sensitive breadth-first search through subsets of π for a satisfiable subset—i.e. starting at π consider π subsets in order of non-increasing cost until finding a satisfiable one. Any satisfying assignment for this set is a MAX-SAT solution provided all weights are non-negative. For negative weights a satisfying assignment for a consistent constraint set may not be a MAX-SAT solution. Although we can always replace a negatively weighted constraint $\langle \phi, c \rangle$ by $\langle \neg \phi, -c \rangle$, $\neg \phi$ may not be Horn, possibly making satisfiability hard. Hence we require non-negative weights.

Since testing satisfiability is efficient for Horn constraints, the time required by the dual approach primarily depends on the number of search nodes we consider. This number is bounded by the number of subsets of π that have a cost greater than $\sigma(O_{1:j}|\Phi)$, which can be exponentially large. Thus, we compute an approximation σ^{τ} to σ by first searching through subsets of π for a maximum of τ steps. We return a solution if one is found and otherwise return an upper-bound to $\sigma(O_{1:j}|\Phi)$ resulting from starting will all constraints and greedily dropping them one at a time in order of increasing cost until a satisfiable subset is found.

Though σ^{τ} will not be correct for all inputs, we know from Section 4, that it need only be a sufficient approximation to guarantee correct sequential inference. When our constraints are nearly sound, σ^{τ} will tend to be sufficient even for small τ. That is, σ^{τ} will equal σ for critical o-sequences. Recall that when $O_{1:j}$ is a critical o-sequence it must be generated by a single state s. Thus, s will satisfy most constraints in $\Gamma^*(O_{1:j}, \Phi)$ and our search need only remove a small number of constraints (the unsatisfied ones) to find a satisfiable subset.

In addition, it is also important to note that the MERGE operation on constraint sets tends to place high weight on the satisfied constraints, which guides the search to remove unsatisfied constraints first. We illustrate this idea with a simple example. Consider an o-sequence $O_{1:j}$ where state proposition p was true throughout. Suppose we have a weighted constraint of the form $\langle b(o) \rightarrow p, c\rangle$ where $b(o)$ is a conjunction of observation tests on observation o. For many of the observations in $O_{1:j}$, $b(o)$ and other constraints that imply p will likely be true resulting in many unit weighted constraints of the form $\langle p, c\rangle$ to be in $\Gamma(O_{1:j}, \Phi)$. All of these unit constraints will be combined in $\Gamma^*(O_{1:j}, \Phi)$ by the MERGE operation giving a single weighted constraint of the form $\langle p, c'\rangle$ where c' is quite large. Alternatively, suppose that state proposition q is not true during $O_{1:j}$. For nearly-sound constraint sets, if we consider rules of the form $b'(o) \rightarrow q$ it is unlikely that many of their bodies $b'(o)$ will be satisfied for observations in $O_{1:j}$. If only a small number of the bodies are satisfied then $\Gamma(O_{1:j}, \Phi)$ will have only a small number of weighted constraints of the form $\langle q, c\rangle$, resulting in $\Gamma^*(O_{1:j}, \Phi)$ placing only a small weight on q. Thus, the dual MAX-SAT approach described above will tend to consider removing the constraint q before removing p as desired.

7. Extending to Relational Processes

In the spirit of knowledge-based model construction (Wellman, Breese, & Goldman, 1992), we extend to relational processes by compiling "relational penalty-logic schemas" to propositional penalty-logic knowledge based and use the ideas from previous sections.

Relational Processes. A sequential inference problem is *relational* when the observation and state spaces $\mathcal{X}$ and $\mathcal{Y}$ are given by specifying a domain set of objects D, a state-predicate set R_s, and an observation-feature set F_o, each having a specified number of arguments. An *observation fact* has the form "$f = v$", where f is an observation feature applied to objects and v is a number. A *state fact* is a predicate from R_s applied to objects. See Example 1 for example facts from our video domain. Observations (states) are finite sets of observation (state) facts, representing all the facts that are true, and $\mathcal{X}$ ($\mathcal{Y}$) contains all such sets. States are restricted to only involve objects that appear in the corresponding observation. We often view relational states as propositional. Given a finite

Table 1: Force-dynamic state predicates R_s (top) and observation facts over our observation features F_o (bottom) for our application.

Attached(x, y)	x supports y by attachment
Grounded(x)	support of x is unknown
Contacts(x, y)	x supports y by contact
Direction(x) = d	x is moving in direction d
Speed(x) = s	x's speed is s
Elevation(x) = e	x's elevation is e
Morph(x) = c	x's shape-change factor is c
Distance(x,y) = d	distance between x and y is d
ΔDist(x,y) = dd	change in distance is dd
Compass(x,y) = c	compass direction of y to x is c
Angle(x,y) = a	angle between x and y is a

$D' \subseteq D$, denote by $\mathcal{Y}[D']$ the propositional state space over n binary variables, one variable for each of the $n = O(|D'|^q)$ state facts involving only objects in D', where q is the maximum state-predicate arity.

Example 2. *In our video domain we infer force-dynamic state sequences from videos of a hand playing with blocks. D contains all hands and blocks we might encounter. There are three force-dynamic state predicates and eight observation features, shown in Table 1. Figure 2 depicts two distinct force-dynamic states. Observations are sets of observation facts calculated for the objects and object pairs based on the object tracker output.*

Relational Horn Constraints. A *state atom* is a state predicate or the relation "$\neq$" applied to variables. An *observation atom* has the form "$(f_1\ r\ f_2)$", where $r \in \{=, \leq\}$, and f_i is a number or an observation feature applied to variables. A *relational Horn constraint* has the form $(\mathit{body} \rightarrow \mathit{head})$, where *body* is a conjunction of state and/or observation atoms, and *head* is a state atom or **false** and may only contain variables that appear in *body*.[3] For example, $(\textsc{Distance}(x, y) \leq 5) \wedge (6 \leq \mathit{Speed}(y)) \rightarrow \textsc{Attached}(x, y)$ is a relational Horn constraint for predicting object attachment based on an observation. A relational Horn constraint ϕ is a schema for propositional constraints. Any way of (consistently) replacing variables in ϕ with objects gives a (propositional) *ground instance* of ϕ. Given a set of objects D', $\textsc{Ground}(\phi, D')$ contains all ground instances with only objects in D'.

Relational Cost Models. A *relational simple-transition model* is a pair $\langle \Phi, K \rangle$, where K is the transition cost and $\Phi = \{\langle \phi_1, c_1 \rangle, \ldots, \langle \phi_v, c_v \rangle\}$ is a Horn penalty-logic schema, where the ϕ_i are relational Horn constraints and the c_i are non-negative costs. Given a relational o-sequence $O_{1:T}$ with objects D', we know that states may only involve facts constructed from D', and thus we need only consider the propositional state space $\mathcal{Y}[D']$. To infer an s-sequence over $\mathcal{Y}[D']$ we use the set of propositional constraints $\Phi_p = \bigcup_{\langle \phi, c \rangle \in \Phi} \bigcup_{\phi' \in \textsc{Ground}(\phi, D')} \langle \phi', c \rangle$ to define an atemporal cost function $C_a(s|o, \Phi_p)$ as in

3. It is straightfoward to extend our definitions and semantics below to allow for constants to appear in the constraints in addition to variables.

```
$\textsc{Perceptron}(\text{TRN}, \{\phi_1, \ldots, \phi_v\}, \tau, M)$
$K \leftarrow 0$; $\vec{C} = \vec{0}$;
repeat  $M$ times,
    for-each $\langle O, S\rangle \in \text{TRN}$
        $\Phi \leftarrow \{\langle \phi_1, c_1\rangle, \ldots, \langle \phi_v, c_v\rangle\}$
        $\hat{S} \leftarrow \text{SSM-DP}(O, \sigma_r^\tau(\cdot|\Phi), K)$
        if  $\hat{S} \neq S$,
            $\vec{C} \leftarrow [\vec{C} + \vec{V}(\hat{S}, O) - \vec{V}(S, O)]^+$
            $K \leftarrow [K + \textsc{Trans}(\hat{S}) - \textsc{Trans}(S)]^+$
return $\langle \vec{C}, K\rangle$
```

Figure 3: Generalized Perceptron Pseudocode. $[\vec{C}]^+ = \vec{C'}$ s.t. $c'_i = \max(0, c_i)$.

Section 6. We then return the lowest-cost $S_{1:T}$ given by SSM-DP$(O_{1:T}, \sigma_r^\tau(\cdot|\Phi), K)$, where $\sigma_r^\tau(O_{1:j}|\Phi) = \sigma^\tau(O_{1:j}|\Phi_p)$ is a relational SSM function with search bound τ. That is σ_r^τ is computed by compilation to a propositional SSM function σ^τ and then using our bounded-search dual MAX-SAT approach.

A naive implementation of this approach can be expensive since Φ_p can be large. Fortunately, in practice, our relational representation allows us to avoid constructing most of the set. It is straightforward to use efficient forward-chaining logical inference to construct $\Gamma^*(O_{1:T}, \Phi_p)$, the input to MAX-SAT, without explicitly constructing Φ_p.

Note that according to the above semantics, the number of propositional constraints that appear in Φ_p for a relational constraint grows with the number of variables in the constraint. This means that relational constraints with more variables have more opportunities to be violated and thus have a bias toward higher costs. Our learning algorithm will automatically takes such biases into account when selecting the costs for relational constraints.

8. Learning a Relational Simple-Transition Model

In this section, we describe our approach for learning relational STMs. Naturally, this same approach can be used to learn propositional STMs which are just special cases with no domain objects. We use a two staged approach where we first learn and/or provide the constraints of the STM and then tune their weights. The approach to acquiring constraints depends on the domain and we give two examples in our experiments. As detailed in Section 9, in one case, we use a combination of classifier learning and human coding, and in another case we automatically learn rules using the relational constraint learner Claudien (De Raedt & Dehaspe, 1997)

Given a set of relational constraints we use a new training set of observation and state sequence pairs to jointly tune the constraint weights and transition-cost K using a variant of Collins' Perceptron algorithm (Collins, 2002) for structured outputs. The algorithm extends Rosenblatt's perceptron for binary labels (Rosenblatt, 1958) to handle structured labels such as sequences, and has convergence and generalization properties similar to Rosenblatt's. The main difference between our variant and Collins' is that we restrict the cost parameters to be non-negative, which is not enforced by Collins' algorithm.

The algorithm requires representing cost using a linear combination of m features, which we do as follows. Given $O_{1:T}$ and $S_{1:T}$ involving just objects in the finite D', and a relational STM $\langle\{\langle\phi_1, c_1\rangle, \ldots, \langle\phi_v, c_v\rangle\}, K\rangle$, the *violation-count feature* of ϕ_i is $V_i(O_{1:T}, S_{1:T}) = \sum_{1\leq i\leq T}\sum_{\phi'\in\text{GROUND}(\phi_i,D')}\delta(\neg\phi'[o_i][s_i])$, i.e. the number of unsatisfied instances of ϕ_i. We let $\vec{V}(O_{1:T}, S_{1:T})$ be the v-dimensional vector of violation-count features and $\vec{C} = [c_1, \ldots, c_v]$ is the weight vector. The *transition count feature* $\text{TRANS}(S_{1:T})$ is equal to the number of state transitions in $S_{1:T}$. It is straightforward to show that with these $v + 1$ features, the STM cost can be represented as $C(S_{1:T}|O_{1:T}) = \vec{V}(O_{1:T}, S_{1:T}) \cdot \vec{C} + \text{TRANS}(S_{1:T}) \cdot K$. For notational convenience we denote the features of an o-sequence/s-sequence pair as $F(O_{1:T}, S_{1:T}) = [\vec{V}(O_{1:T}, S_{1:T}), \text{TRANS}(S_{1:T})]$ and we will often use $w = [\vec{C}, K]$ to denote the composite weight vector. It follows that $C(S_{1:T}|O_{1:T}) = w \cdot F(O_{1:T}, S_{1:T})$.

The algorithm (see Figure 3) cycles through the training data and when an incorrect s-sequence $\hat{S}$ is inferred in place of S, the weights are adjusted to increase the cost of $\hat{S}$ and decrease cost of S. The input is a training set TRN, relational Horn constraints $\{\phi_1, \ldots, \phi_v\}$, an SSM search bound τ, and the number M of iterations. The output is the learned weights $\vec{C}$ and transition cost K. The goal is for the learned STM weights to result in accurate sequential inference when using the search-bound relational SSM function σ_r^τ. Unlike Collins' algorithm we require non-negative weights for inference and thus set a weight to zero if a standard perceptron update would result in a negative value. This variant has not yet been shown to possess the convergence and generalization properties of the unconstrained version for the structured output case. However, this variant has been shown to converge for Rosenblatt's perceptron in the binary classification setting (Amit, Wong, & Campbell, 1989). Below we show that the above algorithm does converge under certain assumptions about τ and the existence of a solution. Following standard practice we first define the margin of a set of weights on a given training set as follows.

Definition 2 (Margin). *A weight vector w has margin δ on o-sequence, s-sequence pair $\langle O_{1:T}, S_{1:T}\rangle$ if for all other s-sequences $S'_{1:T}$, $w \cdot F(O_{1:T}, S'_{1:T}) \geq w \cdot F(O_{1:T}, S_{1:T}) + \delta$, i.e. $C(S'_{1:T}|O_{1:T}) \geq C(S_{1:T}|O_{1:T}) + \delta$. The weight vector w has margin δ on a training set of o-sequence, s-sequence pairs if it has a margin of δ on each pair.*

The following shows that if τ is selected large enough to guarantee sufficient SSM and there is a non-negative set of weights with unit norm and a positive margin, then the algorithm is guaranteed to converge in a finite number of iterations. Note that since it is always possible to uniformly scale the weights without changing the inference results, the focus on unit norm weights is not limiting. The proof is straightforward and can be seen as an extension of Collins' result to non-negativity constraints or a extension of (Amit et al., 1989) to structured outputs.

Proposition 4. *Assume that τ is large enough such that for all sets of weights, σ_r^τ allows sufficient SSM. If there exists a non-negative weight vector w such that $\|w\| = 1$ and w has margin δ on the training data, then the algorithm in Figure 3 converges to a non-negative weight vector that correctly solve all of the training problems after committing no more than*

$\left(\frac{R}{\delta}\right)^2$ errors on the training set. Here R is a constant such that for any $O_{1:T}, S_{1:T}, S'_{1:T}$, $\|F(O_{1:T}, S_{1:T}) - F(O_{1:T}, S'_{1:T})\| \leq R$.

Proof. Let w_k be the weights that are obtained by the algorithm after the k'th mistake. Also, let $O_{1:T}$ and $S_{1:T}$ be the o-sequence and target s-sequence on which the k'th mistake was made, and let $S'_{1:T}$ be the incorrectly predicted s-sequence for $O_{1:T}$ using weights w_{k-1}. We have that $w_0 = 0$ and $w_{k+1} = [w_k + F' - F^*]^+$, where $F' = F(O_{1:T}, S'_{1:T})$ and $F^* = F(O_{1:T}, S_{1:T})$.

The proposition follows from two inequalities $\|w_{k+1}\|^2 \leq kR^2$ and $w \cdot w_{k+1} \geq k\delta$, which are derived below. Given these we observe that

$$k\delta \leq w \cdot w_{k+1} \leq \|w\|\|w_{k+1}\| = \|w_{k+1}\| \leq \sqrt{k}R$$

from which we derive $k \leq \left(\frac{R}{\delta}\right)^2$, showing that the number of mistakes is bounded as stated in the proposition.

To complete the proof we derive the above two inequalities. First we bound $\|w_{k+1}\|^2$ as follows.

$$\begin{aligned}
\|w_{k+1}\|^2 &= \|[w_k + F' - F^*]^+\|^2 \\
&\leq \|w_k + F' - F^*\|^2 \\
&= \|w_k\|^2 + 2w_k \cdot [F' - F^*] + \|F' - F^*\|^2 \\
&\leq \|w_k\|^2 + 2w_k \cdot [F' - F^*] + R^2 \\
&\leq \|w_k\|^2 + R^2
\end{aligned}$$

Using the fact that $w_0 = 0$ and the above inequality we have by induction that $\|w_{k+1}\|^2 \leq kR^2$. The first inequality follows from the fact that for any vector v, $\|[v]^+\|^2 \leq \|v\|^2$. The third inequality follows by assumption from the proposition statement. The final inequality follows from the fact that we assume σ^τ is a sufficient SSM approximation. In particular, since σ^τ is sufficient, it follows from Proposition 3 that $S'_{1:T}$ is a minimal cost sequence for $O_{1:T}$ given weights w_k. Since the costs of $S'_{1:T}$ and $S^*_{1:T}$ under w_k are $w_k \cdot F'$ and $w_k \cdot F^*$ respectively, we have that $w_k \cdot F' \leq w_k \cdot F^*$ which implies the final inequality.

Next we derive the lower bound on $w \cdot w_{k+1}$, where w is the unit length weight vector that achieves margin δ on the training set. First notice that the update equation can be rewritten as,

$$w_{k+1} = [w_k + F' - F^*]^+ = w_k + F' - F^* + [-(w_k + F' - F^*)]^+$$

With this we can derive the following bound.

$$\begin{aligned}
w \cdot w_{k+1} &= w \cdot w_k + w \cdot [F' - F^*] + w \cdot [-(w_k + F' - F^*)]^+ \\
&\geq w \cdot w_k + w \cdot [F' - F^*] \\
&\geq w \cdot w_k + \delta
\end{aligned}$$

By induction it follows that $w \cdot w_{k+1} \geq k\delta$. The first inequality follows from the fact that by definition all components of $[v]^+$ are non-negative and the components of w are non-negative by assumption. The second inequality follows from the margin assumption about weight vector w. □

9. Experimental Results

We present experimental results in two domains: 1) wearable webcam location tracking, where we use a propositional simple-transition model, and 2) force-dynamic state interpretation from video where we use the full relational simple-transition model.

9.1 Wearable Webcam Location Tracking

In this section, we apply our simple-transition model to the problem of location tracking based on data captured by a wearable webcam. We use the dataset collected in Torralba et al. (2003) using a head-mounted 120x160 pixel color web-cam. The dataset includes 17 sequences, each arising from the movement of a subject through an environment under realistic conditions. Each sequence goes through a series of locations, e.g. particular offices, streets, corridors, elevators, labs, kitchens, etc, for a total of 63 distinct locations. The goal is to process the sequences of video frames to infer the sequence of locations that were traversed. Thus, here, the hidden states correspond to location names and the raw observations correspond to 120x160 pixel video frames. Rather than working with raw pixel values as the observations, Torralba et al. (2003) process the video frames arriving at an 80-dimensional real-valued feature vector representation that describes global properties of video frames. These feature vectors are then treated as the observations associated with each time point. All of our experiments use the filter-bank feature set from Torralba et al. (2003). Note that the state space of this problem is small enough that Viterbi could be applied. The main aim of our experiments with this data set is to investigate the effectiveness of the simple-transition model on a complex real-world problem where it is typical to utilize more complex models.

Propositional Representation. In order to utilize our previously described propositional representation, we first re-represent the 80-dimensional feature vectors as a set of propositional variables $\{x_1, \ldots, x_{63}\}$, one proposition for each location of the world. We use a learned classifier to transform the original numeric observations to the propositional representation. To do this we followed the approach used in (Torralba et al., 2003) and randomly sampled 100 observation vectors for each state in the training data. These prototype vectors were then used to create a Parzen window density estimator over observation vectors conditioned on the state. These estimators were then used to define a classifier from video frames to states, by predicting the class that was most likely. Finally given this classifier we define x_i to be true for a video frame if and only if the classifier predicts state i for the frame. We treat the assignment of truth values to these propositions as a propositional observation where each proposition intuitively serves as a specialized noisy detector for a particular location of the world. As we will see in the experiments, these propositions are very noisy and achieve poor accuracy in predicting the actual location from single frames. The state space for this problem is also represented by a set of propositions $\{y_1, \ldots, y_{63}\}$, one proposition for each state of the world, where y_i is active if and only if the true state of the world is i. The goal of sequential inference will be to infer an accurate state sequence given the sequence of propositional observations.

Constraint Set. Recall that our penalty-logic simple-transition model is defined by a set of weighted rules defining the atemporal cost function and a transition cost, and that our weight learning algorithm assumes a set of rules be provided. For this experiment, we simply provided the following two sets of Horn rules. The first set

$$\{x_i \rightarrow y_j \ : 1 \leq i \leq n, 1 \leq j \leq n\}$$

allows the learner to learn the cost of predicting y_j when x_i is true. The second set of rules

$$\{\neg y_i \vee \neg y_j : i \neq j\}$$

involve only state propositions and constraints the state space to having only a single proposition true for any give time point. Since we know that the second set of rules represent hard constraints, i.e. only one location is present at once, we fix their weight values to effectively be infinity. Note that the constraints in the first set are not nearly-sound in this case, indicating that the SSM inference method described previously may not be effective. However, in this domain it is not hard to prove that the rigid structure of the state proposition constraints, mainly that exactly one state proposition must be true, allows for the greedy hill-climbing approach to return optimal MAX-SAT solutions. Thus, for this experiment we solve SSM using pure greedy hill climbing, i.e. $\tau = 0$. The execution time for this inference is almost instantaneous.

Given this set of rules we used the Perceptron algorithm to learn a simple-transition model—in particular learning the transition cost and the weights of the first constraint set above. For comparison we also used the Perceptron algorithm to learn a full HMM model following (Collins, 2002). This second approach is similar to that used by Torralba et al. (2003) who used a full HMM model but with generative training and with a Parzen density estimator used for the probabilistic observation model.

The only difference between our discriminatively trained STM and full HMM models is that the full HMM model learns a cost for each possible state transition, for a total of 63x63 different costs, whereas the STM learns a single transition cost parameter. In this domain, there is clearly first-order Markovian transition structure among state/location transitions since some locations are not immediately reachable by others. Intuitively, this suggests that one should use the full HMM transition matrix rather than the STM model, which clearly ignores the state-transition structure. However, in this domain, it is also the case that subjects stay in locations for many observations (e.g. it takes time to walk from one location to another) and that considering all those observations together can often give a good indication of the location. This suggests, that perhaps the location transition structure is not critical for high accuracy and that an STM may provide good results. In such situations, it is preferable to use an STM since there are many fewer parameters to tune and hence over-fitting to a small training set is less of a problem. Our results below demonstrate that this is the case in this domain.

Results. Following Torralba et al. (2003) we evaluate our approach via leave-one-sequence-out cross-validation using the 17 sequences in the dataset. For each sequence we recorded the percent prediction error after training on all other sequences and the number of Perceptron iterations required to reach maximum performance. Here prediction error is measured as the percentage of mispredicted locations in a given sequence. We also report the average and median percent error across all sequences. Table 2 gives these results.

The table gives results for three different models. First we give results for the Zeroth Order model, which corresponds to the error rate that can be achieved by using the prediction of the observation model alone with no transition model. That is, predictions are made at each time point in an i.i.d. fashion. The STM column gives results for the simple-transition model described above. The Full First Order column gives results for the full transition matrix model described above. Comparing the zeroth order model to the sequential models, we see that for this dataset there are significant gains when using a transition model in place of simply making i.i.d. predictions. We also see that for most sequences the STM significantly outperforms the full first-order model in terms of both prediction error and training iterations. Given that the full first-order model has significantly more parameters to tune, it is not unexpected that its training time is significantly longer. However, we also see that the use of the more complex model, appears to lead to over-fitting given our fixed dataset compared to using the much simpler STM. Thus, for this dataset the STMs simple-transition bias appears to pay off due to reduced variance.

The STM performs similarly to the full first-order model developed in Torralba et al. (2003)[4], achieving a median error of 26% compared to median error of approximately 28% obtained by Torralba et al. (2003). The model in that work utilized priors and a likelihood-rescaling parameter, tuned via cross-validation, to effectively avoid the over-fitting observed with our full first-order model, which did not include any form of regularization. In contrast, our STM provides an alternative, and arguably simpler approach, to avoiding overfitting, by utilizing a much simpler model, while maintaining adequate expressive power.

9.2 Force-Dynamic State Inference from Video

Here we apply our relational STM model to the problem of force-dynamic state inference from real video. The LEONARD system (Siskind, 2001) uses these states to recognize events, such as "a hand picked up a block" (see Figure 1). Recently (Fern & Givan, 2004) developed a trainable system for this problem using the *forward greedy merge (FGM)* algorithm, which outperformed prior techniques. FGM also utilizes Horn constraints, but assumes that they are sound, rather than nearly sound. Since this is not true for CLAUDIEN-learned constraints, which were also used by FGM, that work utilized two ad-hoc steps to improve performance: 1) A *constraint pruning* procedure was developed that searched for a small set of constraints that appeared "sufficient" for the training data, thus reducing the chance for a constraint violation. 2) *Sequence-cleaning (SC) preprocessing* was a step that removes observations where certain types of constraint violations are "detected". See (Fern & Givan, 2004) for details. While these steps allowed for good performance, the soundness assumption limits the approach's applicability, as such preprocessing will not always be effective. The original motivation for the work in this paper was to develop a "softened" more robust framework for utilizing nearly-sound constraints for sequential inference.

Procedure. Our corpus of 210 videos from (Siskind, 2003) contains 7 event types (30 movies each) involving a hand playing with up to three blocks (e.g. assembling a tower), comprising a total of 11946 video frames. We use a training set of 21 hand-labeled movies, 3 of each event type from (Fern & Givan, 2004). Each video frame was processed resulting in a relational observation sequence involving the observation predicates described in Table

4. This is inferred by comparing our median value with the 100% recall point of the "Filter Bank" curve from Figure 4a in Torralba et al. (2003).

Table 2: % error on web-cam data set for each of the 17 sequences with average and median results. Zeroth Order corresponds to the prediction obtained by using the observation model alone with no transition model. STM corresponds to the simple-transition model. Full First Order corresponds to a transition model with a full 63x63 transition matrix. The columns labeled iterations give the number of iterations of the Perceptron algorithm through the training set required to reach maximum performance.

	Zeroth Order	STM		Full First Order	
Sequence	% Error	% Error	Iterations	% Error	Iterations
1	82	24	29	50	763
2	78	29	45	36	3171
3	83	33	40	56	2421
4	88	49	21	74	2558
5	87	29	118	79	2899
6	85	33	37	65	3683
7	85	35	439	62	4082
8	81	60	131	58	3822
9	80	20	16	34	3645
10	76	22	25	32	860
11	85	47	136	42	1259
12	74	17	26	29	2568
13	84	24	38	61	2004
14	79	19	11	57	2619
15	69	18	31	31	3015
16	67	26	56	36	3012
17	72	26	89	46	1537
Average % Error	80	30		50	
Median % Error	81	26		50	

1. The label for each frame gives the relational force-dynamic state of the frame in terms of the three force-dynamic predicates shown in Table 1.

In this domain, the eventual use of the force-dynamic models is for activity recognition and as such the utility of an inferred s-sequence S (giving a sequence of relational force-dynamic states) primarily derives from the sequence of distinct states, rather than identifying the exact state transition points. Indeed, in our video domain, the exact locations of state transitions are often ambiguous (as judged by a human) and unimportant for recognizing activity—e.g. in Figure 1 it is unimportant to know in precisely which frame the force-dynamic state transition occurs. Furthermore, LEONARD is able to accurately infer event-types from force-dynamic sequences provided that the sequence of distinct states is correct, regardless of the precise state transition points. In this light, we define our accuracy measure for this domain as follows. Let $\textsc{Compress}(S)$ denote the sequence obtained by removing consecutive repetitions in S. For example, $\textsc{Compress}(a, a, a, b, b, a, c, c) = a, b, a, c$. Our goal is to map an o-sequence O to an s-sequence S such that $\textsc{Compress}(S)$ is the distinct sequence of states that generated O. Our measure of accuracy is simply the percentage of sequences for which the compressed force-dynamic state is correctly inferred. This measure is directly related to the event recognition accuracy achievable by LEONARD using the learned force-dynamic models. To measure accuracy using this metric we labeled the 189 videos outside of the training set by their compressed s-sequences, which is considerably less time-consuming than labeling each individual frame as required for the training data. We then measured the error rate relative to these compressed s-sequences.

Rule Learning. In this domain, we learned Horn rules using the relational rule mining CLAUDIEN (De Raedt & Dehaspe, 1997). CLAUDIEN takes as input a set of first-order models and a declarative language bias that specifies a space of first-order clauses to consider. It outputs the most general clauses consistent with the language bias that satisfy specified coverage and accuracy constraints. We used 7 videos from our training set to generate training data for CLAUDIEN. This number was selected to strike a balance between data coverage (one per event type) and runtime of CLAUDIEN which took nearly a week. For each video frame from the videos we created a model that specified the observation and state ground facts for that frame. The entire set of models was then fed to CLAUDIEN with a language bias restricting Horn rules to only allow state predicates in the head (empty heads were also allowed). This restriction was selected so as to include the classes of rules that we expect to be most useful, while reducing the search space enough to allow for reasonable runtimes. We used an accuracy constraint of 100% (i.e. the rule must not be violated in the training data) and a coverage constraint of 10. The result was a set of 479 Horn rules which we used to define our relational STM model. Note that although we used an accuracy constraint of 100%, the rules were not perfect for models outside of the 7 training movies, though each rule was rarely violated.

Evaluated Systems. We learned STM models using the Perceptron algorithm to learn the transition cost and weights for the CLAUDIEN discovered rules. For inference, we used the dual Max-SAT procedure with three SSM search bounds $\tau = 0, 100, 1000$ combined with the PRUNED-SSM-DP. This resulted in three different STM models, each tuned for a particular search bound. During testing of each model the same τ was used for inference as was used during training. All 21 training models, were used for weight learning. In each case we ran the Perceptron algorithm for 100 iterations noting that fewer than 10 iterations were required to reach maximum performance. Note that we conducted preliminary tests using (unpruned) SSM-DP for inference and found that for our data PRUMED-SSM-DP was at least an order of magnitude more efficient while producing identical results, which was critical for reasonable training and evaluation times.

We compare against FGM used with and without sequence cleaning and always with pruning, since without pruning results are very poor. We also compare against a system that is closer to mainstream relational graphical modeling techniques, which uses the counting-transition model of Section 4, and WALKMAXSAT (Jiang et al., 1995) for approximate inference. The model is identical to our relational STMs, using the same CLAUDIEN-learned constraints, except that it has a non-simple-transition structure. For sequential inference we construct a single large MAX-SAT problem, with one variable for each state fact at each time-step[5], corresponding to the counting-transition model and use WALKMAXSAT to find

5. Actually, we first apply the compress operation to the input o-sequence to arrive at a shorter sequence and then introduce one variable for each state fact for each compressed sequence index. This has the effect of creating smaller, but equivalent, MAX-SAT problems.

Table 3: % error on test movies across training iterations. The last column gives frames processed per second (FPS) by the best model in each row on our video corpus.

	Iterations								FPS
	1	2	3	4	5	6	7	8	
SSM-DP(0)	94	6.3	6.9	5.8	4.8	12	17	3.7	15
SSM-DP(100)	93	3.2	3.7	3.7	3.2	10	21	8.5	14
SSM-DP(1000)	93	3.2	3.7	3.7	3.2	10	21	8.5	8
WalkMaxSat	95	47	50	60	49	50	45	52	1.4
FGM / FGM+SC	15 / 3.2								29

an approximate solution.[6] We tune the model's weights using the perceptron algorithm with WalkMaxSat (rather than SSM-DP) for inference. This model can be viewed as a discriminately trained relational Markov network (Taskar, Abbeel, & Koller, 2002) using a penalty-logic-based representation. More recent work has also considered applying the Perceptron algorithm to this style of model under the name Markov Logic (Singla & Domingos, 2005).

Performance Across Iterations. Table 3 shows, for each training set, the testing error of our learned STMs for $\tau = 0, 100, 1000$ (shown as SSM-DP(τ)), over the first eight perceptron iterations. There is always a rapid improvement after iteration one, followed by a period of fluctuating or constant performance. A major reason for this improvement is that certain rules actually correspond to hard constraints of the domain (e.g. a block can not be attached to two other blocks) and their cost increases, forcing the MAX-SAT solutions to satisfy them. The results fluctuate out to iteration 100 (not shown), but never improve over the best performance shown. Such fluctuating behavior is not uncommon for the Perceptron algorithm (Collins, 2002). In practice, one could use cross-validation to select a good model, or consider the use of "weight averaging" (Collins, 2002) to produce more stable learning curves.[7]

Bounding Search. The search bound $\tau = 0$ means that SSM is (approximately) solved via search-free hill-climbing. The results show that even wihtout search we are able to learn weights that perform quite well achieving an error of 3.7%. Increasing the search bound to 100 improves performance with respect to the best model across iterations, but moving to 1000 did not improve results. The last column shows the frames processed per second when inferring states for our corpus. Inference time apparently does not increase linearly with τ, indicating that the SSM search typically end much before reaching the bound. Increasing τ to 10,000, doubles inference time and learning time, but neither improves nor hurts results.

Other Techniques. We significantly outperform WalkMaxSat (further iterations out to 1000 did not help), which we allowed a generous search time, using search cutoff 10^6 with 10^3 random restarts (other settings did not improve). Further experiments suggest that

6. We also conducted a similar experiment where we encoded the simple-transition model into the MAX-SAT problem and got similar results to the more general counting-transition model.

7. Runtime of Claudien (interpreted Prolog) was about six days. Thus, it was not feasible to average results across many training sets to get smoother training curves. In another full experiment (21 movies), we observed similar (slightly better) performance for SSM-DP, and similar relative performance to the other systems.

WalkMaxSat is simply not scaling well for our longer sequences (the number of variables is linear in the sequence length). In one experiment, we used learned STM weights and encoded the model as a large MAX-SAT problem and used WalkMaxSat for inference. WalkMaxSat was able to correctly infer s-sequences for many of the shortest videos, but it performed very poorly for long videos. This suggests the poor performance is primarily due to the ineffectiveness of this general-purpose inference technique for our models. It is possible that improved general-purpose MAX-SAT procedures may fair better on our problems. However, our results show that for STMs such techniques are not required as the SSM problem decomposition allows for a relatively simple and generic approach to be effective.

FGM without sequence cleaning (shown as FGM) is significantly worse than STMs (which never use cleaning). We yield equal performance when sequence cleaning is used for FGM. These results indicate that our penalty-logic based framework provides an automatic and robust way to utilizing a large set of nearly-sound logical constraints. The FGM approach require careful ad-hoc processing of the constraint set and sequence cleaning to yield the same performance. FGM is faster, close to frame rate. A reimplementation of our LISP prototype will likely achieve frame rate.

10. Related Work

Segment models (Ostendorf et al., 1996) generalize first-order Markov models by allowing for arbitrary distributions on the durations of states and on the sequences of observations given an interval/segment where a state is active. As noted in Section 4, our approach to STM inference is similar to algorithms for segment model inference. We are not aware, however, of prior work that has leveraged or noted the SSM reduction for large state spaces. Perhaps this is because prior segment modeling work typically utilizes (non-simple) transition structure (perhaps sometimes unnecessarily) and is typically applied to small state spaces that can be enumerated during inference.

There are a variety of approximate sequential inference approaches that have been developed for problems involving large state and observation spaces. One generic approach is to utilize beam search within Viterbi inference, resulting in complexity that does not depend on the size of the state space, with the potential cost of pruning away optimal solutions. Some examples of this approach with integrated learning algorithms include Collins and Roark (2004), Daume III and Marcu (2005), Xu and Fern (2007). Related to these approaches are forward probabilistic filtering approaches such as particle filtering (Doucet, de Freitas, & Gordon, 2001) and the Boyen-Koller algorithm (Boyen & Koller, 1998), which maintain and propagate compact, approximate representations of the state distribution at each time step given the previous observations. An interesting direction of future work is to compare these approaches which utilize approximate inference on expressive models to approaches that consider learning restricted models that support efficient inference such as our STM approach.

Our "model schema" approach for handling relational data is now standard in probabilistic modeling (De Raedt & Kersting, 2004). Our overall relational STM model can be viewed as a special case of relational Markov networks (RMNs) (Taskar et al., 2002). A relational STM can be viewed as defining a log-linear conditional RMN, where the RMN "clique feature templates" correspond to relational Horn constraints and a state transition template. RMNs are commonly trained by optimizing the conditional likelihood or a related measure. Here we have chosen to drop the probabilistic interpretation during training in favor of the much simpler Perceptron algorithm, which does not require the computation of clique expectations. Note, if desired, one may still interpret the learned cost functions as defining a log-linear probabilistic model, though with unclear semantics. RMNs are very general and thus prior techniques do not fully exploit the structure of relational STMs. Instead RMN-like proposals rely on general-purpose approximate inference (e.g. belief propagation), with unclear practical implications. Likewise dynamic probabilistic relational models (Sanghai, Domingos, & Weld, 2003) provide a generic schema-based extension of dynamic Bayesian networks (Murphy, 2002) specialized to relational sequence data. Again the generality precludes leveraging the STM structure.

Recent work on logical hidden Markov models (LOHMMs) (Kersting, DeRaedt, & Raiko, 2006) considers extending HMMs to deal with states and observations represented as first-order logical atoms, possibly with structured terms. This represents a significant extension to the range of problems approachable by HMM style approaches as the structure in the atoms can be leveraged for improved generalization and inference. However, LOHMMs do not directly apply to the types of relational processes described in this paper, since our processes use sets of logical atoms to describe states and observations. One benefit of considering only single atoms is that it allowed for a more straightforward lifting of HMMs to the relational setting. However, the cost of the restriction is loss of expressive power for states and observations. Our model, rather, can handle sets of atoms, but relies on a highly simplified transition model compared to LOHMMs. Thus, our model is not appropriate for many processes where LOHMMs might be expected to work well. Considering a tractable combination of these two models is an interesting future direction.

The use of weighted constraints for defining cost functions is common in the area of constraint optimization and has been found to be an effective way for dealing with preferences or inconsistency. There has been much less work on learning sets of weighted constraints for the purpose of structured classification as in our application. Derthick (Derthick, 1990) described how it is relatively straightforward to employ probabilistic methods for learning penalty-logic models, however, this idea was not implemented. To the best of our knowledge, prior to our original work (Fern, 2004) there were no implemented and evaluated approaches for learning penalty-logic-based models. In concurrent work, (Richardson & Domingos, 2006) learned such models in a probabilistic setting under the name Markov Logic Networks. Follow-up work (Singla & Domingos, 2005) also considered training via the Perceptron algorithm.

11. Summary

This paper introduced the penalty-logic simple-transition model as a way of leveraging nearly-sound logical constraints for sequential inference in factored state spaces. First, we showed that inference in simple-transition models can be efficiently reduced to the single-state minimization problem. Furthermore, we showed that even slight extensions to our model render such reductions impossible unless P=NP. Second, we introduced a penalty-logic representation for atemporal cost models and showed how to leverage this representation for practically efficient SSM. Third, we showed how to learn the weights for our model using a straightforward instantiation of a sign-constrained structured Perceptron algorithm. Results in two domains indicate that our model can have advantages over more general ones when applicable. A major thrust of future work involves applying our model to new domains, generalizing the atemporal portion to handle richer constraint languages, and investigating tractable generalizations to richer transition structures.

Acknowledgments

This work was supported by NSF grant IIS-0546867.

References

Amit, D., Wong, K., & Campbell, C. (1989). Perceptron learning with sign-constrained weights. *Journal of Physics A: Mathematical and General*, *22*, 2039–2045.

Boyen, X., & Koller, D. (1998). Tractable inference for complex stochastic processes. In *Conference on Uncertainty in Artificial Intelligence*.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with the perceptron algorithm. In *Conf. on Empirical Methods in NLP*.

Collins, M., & Roark, B. (2004). Incremental parsing with the perceptron algorithm. In *Proc. Annual Meeting for the Assoc. for Computational Linguistics*.

Daume III, H., & Marcu, D. (2005). Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*.

De Raedt, L., & Dehaspe, L. (1997). Clausal discovery. *Machine Learning*, *26*, 99–146.

De Raedt, L., & Kersting, K. (2004). Probabilistic logic learning. *ACM-SIGKDD Explor.*, *5*.

Dechter, R. (1999). Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, *113*(1-2).

Derthick, M. (1990). Mundane reasoning by settling on a plausible model. *Artificial Intelligence*, *46*(1-2).

Doucet, A., de Freitas, N., & Gordon, N. (Eds.). (2001). *Sequential Monte Carlo Methods in Practice*. Springer.

Dragunov, A., Dietterich, T., Johnsrude, K., McLaughlin, M., Li, L., & Herlocker, J. (2005). Tasktracer: A desktop environment to support multi-tasking knowledge workers. In *International Conference on Intelligent User Interfaces*.

Felzenszwalb, P., Huttenlocher, D., & Kleinberg, J. (2003). Fast algorithms for large state space hmm with applications to web usage analysis. In *Advances in Neural Information Processing Systems 16*.

Fern, A., & Givan, R. (2004). Relational sequential inference with reliable observations. In *ICML*.

Fern, A. (2004). *Learning Models and Formulas of a Temporal Event Logic*. Ph.D. thesis, Purdue University, School of Electrical and Computer Engineering.

Fern, A. (2005). A simple-transition model for relational sequences. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Forney, G. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, *61*(3), 268–278.

Jaumard, B., & Simeone, B. (1988). On the complexity of the maximum satisfiability problem for Horn formulas. *Information Processing Letters*, *26*, 1–4.

Jiang, Y., Kautz, H., & Selman, B. (1995). Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. In *Joint Workshop on AI and OR*.

Kersting, K., DeRaedt, L., & Raiko, T. (2006). Logical hidden markov models. *Journal of Artificial Intelligence Research*, *25*, 425–456.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, pp. 282–289.

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, UC Berkeley, Computer Science.

Ostendorf, M., Digalakis, V., & Kimball, O. (1996). From HMMs to segment models: a unified view of stochastic modeling for speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, *4*, 360–378.

Papadimitriou, C. H. (1995). *Computational Complexity*. Addison-Wesley Publishing.

Pinkas, G. (1991). Propositional non-monotonic reasoning and inconsistency in symmetric neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Pinkas, G. (1995). Reasoning, nonmonotonicity and learning in connectionist networks that capture propositional knowledge. *Artificial Intelligence*, *77*(2).

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*(1–2), 107–136.

Rosenblatt, F. (1958). The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–408.

Sanghai, S., Domingos, P., & Weld, D. (2003). Dynamic probabilistic relational models. In *International Joint Conference on Artificial Intelligence*.

Shen, J., Li, L., & Dietterich, T. (2007). Real-time detection of task switches of desktop users. In *International Joint Conference on Artificial Intelligence*.

Singla, P., & Domingos, P. (2005). Discriminative training of markov logic networks. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*.

Siskind, J. M. (2001). Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *Journal of Artificial Intelligence Research*, *15*, 31–90.

Siskind, J. M. (2003). Reconstructing force-dynamic models from video sequences. *Journal of Artificial Intelligence*, *150*(1–2), 91–154.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *Conference on Uncertainty in Artificial Intelligence*.

Torralba, A., Murphy, K., Freeman, W., & Rubin, M. (2003). Context-based vision system for place and object recognition. In *International Conference on Computer Vision*.

Veksler, O. (1999). *Efficient Graph-based Energy Minimization Methods in Computer Vision*. Ph.D. thesis, Cornell University, Computer Science.

Wellman, M., Breese, J., & Goldman, R. (1992). From knowledge bases to decision models. *Knowledge Engineering Review*, *5*.

Xu, Y., & Fern, A. (2007). On learning linear ranking functions for beam search. In *ICML*.

Appendix A. Omitted Proofs

Proposition 1. *Given a STM C and o-sequence $O_{1:T}$, for any $0 \leq j < T$ we have that*

$$C^*(O_{1:T}) \leq C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})$$

with equality if and only if there is a member of $C^*_w(O_{1:T})$ with final transition at j.

Proof. For the case of $j = 0$ the proposition is trivial. We first show the inequality for the case of $j > 0$. Consider the s-sequence $S'_{1:T}$ where $S'_{1:j} \in C^*_w(O_{1:j})$ and each state in $S'_{j+1:T}$ is $\sigma_w(O_{j+1:T})$. If s'_j is not equal to $\sigma_w(O_{j+1:T})$, i.e. $S'_{1:T}$ has a final transition at j, then it is easy to show that $C(S'_{1:T}|O_{1:T}) = C^*(O_{1:j}) + K + \sigma(O_{j+1:T})$. Otherwise we have that $s'_j = \sigma_w(O_{j+1:T})$ which gives $C(S'_{1:T}|O_{1:T}) = C^*(O_{1:j}) + \sigma(O_{j+1:T})$. This shows that $C(S'_{1:T}|O_{1:T}) \leq C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})$. Combining this inequality with the fact that $C^*(O_{1:T}) \leq C(S'_{1:T}|O_{1:T})$ we get that $C^*(O_{1:T}) \leq C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})$, as desired.

We now show that equality holds for $j > 0$ if and only if there exists a member of $C^*_w(O_{1:T})$ with final transition at j. To show the backward direction assume that there is some member $S^*_{1:T}$ of $C^*_w(O_{1:T})$ with final transition at j. Note that for any s-sequence $S_{1:T}$ with a final transition at j we have that

$$\begin{aligned} C(S_{1:T}|O_{1:T}) &= C(S_{1:j}|O_{1:j}) + K \cdot \delta(j > 0) + \sum_{j+1 \leq i \leq T} C_a(s_{j+1}|o_i) \\ &\geq C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T}) \end{aligned}$$

where the first line follows since we know the final state transition occurs between j and $j+1$. Thus, we get that $C^*(O_{1:T}) = C(S^*_{1:T}|O_{1:T}) \geq C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})$. This combined with the inequality from the first part of the proof gives that $C^*(O_{1:T}) = C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})$, as desired.

To show the forward direction assume that $C^*(O_{1:T}) = C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})$ and consider the s-sequence $S'_{1:T}$ where $S'_{1:j} \in C_w(O_{1:j})$ and $s'_i = \sigma_w(O_{j+1:T})$ for $j+1 \leq i \leq T$. For the sake of contradiction assume that $S'_{1:T}$ does not have a final transition at j, i.e. the final transition must occur before j. In this case we have that

$C(S'_{1:T}) = C^*(O_{1:j}) + \sigma(O_{j+1:T})$. Using this expression along with the facts that $K > 0$ for STMs and $j > 0$ we get that,

$$\begin{aligned} C^*(O_{1:T}) &\leq C(S'_{1:T}|O_{1:T}) \\ &= C^*(O_{1:j}) + \sigma(O_{j+1:T}) \\ &< C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T}) \end{aligned}$$

This shows that $C^*(O_{1:T}) \neq C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})$, which contradicts our original assumption. Thus, $S'_{1:T}$ must have a final transition at j. Thus we know that $C(S'_{1:T}|O_{1:T}) = C^*(O_{1:j}) + K \cdot \delta(j > 0) + \sigma(O_{j+1:T})$ which equals $C^*(O_{1:T})$ by assumption. Thus $S'_{1:T} \in C^*_w(O_{1:T})$ and we have shown that some s-sequence in $C^*_w(O_{1:T})$ has a final transition at j which completes the proof. □

Proposition 2. *Given a STM C and an o-sequence $O_{1:T}$, let the sets L_t be computed as described above. For any $1 \leq t < T$ and any $0 \leq j < t$, if $j \in L_t$ and $j \notin L_{t+1}$ (i.e. j was removed on iteration t), then there is a $j' \in L_{t+1}$ such that for any $t' > t$ if $C^*_w(O_{1:t'})$ contains an s-sequence with final transition at j then it also contains an s-sequence with final transition at j'.*

Proof. We use induction on the iteration number t. For the base case of $t = 1$, we have that $L_1 = \{0\}$ and it is easy to see that 0 will also be in L_2 (i.e. L_{t+1}) since $C^*(O_{1:1}) = \sigma(O_{1:1})$ which does not satisfy the condition for pruning $j = 0$. Thus, the proposition is trivially satisfied for the base case of $t = 1$.

For the inductive case of $t > 1$ assume that for some $0 \leq j < t$, $j \in L_t$ and $j \notin L_{t+1}$. Note that since j was pruned from L_t when constructing L_{t+1} we know that $C^*(O_{1:t}) \leq C^*(O_{1:j}) + \sigma(O_{j+1:t}) - K \cdot \delta(j = 0)$.

Let j' be a final transition point for some sequence in $C^*(O_{1:t})$. We know from the inductive hypothesis that at least one such index must be in L_t, since otherwise the pruning would have been unsound for some $t' < t$, which violates the inductive hypothesis. Furthermore we know that j' will not be pruned from L_t and thus must be in L_{t+1} since by Proposition 1 we have that $C^*(O_{1:t}) = C^*(O_{1:j'}) + K \cdot \delta(j' > 0) + \sigma(O_{j'+1:t})$, which does not satisfy the pruning constraint.

The above argument shows that j' will be in L_{t+1} along with t, which is included in L_{t+1} by definition. To complete the proof we show that one of these indices will be at least as good as using j as a final transition point in future iterations. To see this consider some $t' > t$ and assume that j is a final transition point of some sequence in $C^*_w(O_{1:t'})$. We show that either t or j' must also be a final transition point for some member of $C^*_w(O_{1:t'})$.

Let $S'_{1:t'}$ be a sequence such that $S'_{1:t} \in C^*(O_{1:t})$ and for each $i > t$ $s'_i = \sigma_w(O_{t+1:t'})$. Note that $S'_{1:t'}$ has a final transition at either t or j'. The following derivation shows that $S'_{1:t'}$ is an optimal s-sequence.

$$\begin{aligned}
C(S'_{1:t'}|O_{1:t'}) &\leq C^*(O_{1:t}) + K + \sigma(O_{t+1:t'}) \\
&\leq C^*(O_{1:j}) + \sigma(O_{j+1:t}) - K \cdot \delta(j=0) + K + \sigma(O_{t+1:t'}) \\
&= C^*(O_{1:j}) + \sigma(O_{j+1:t}) + K \cdot \delta(j>0) + \sigma(O_{t+1:t'}) \\
&\leq C^*(O_{1:j}) + K \cdot \delta(j>0) + \sigma(O_{j+1:t'}) \\
&= C^*(O_{1:t'})
\end{aligned}$$

Here the first line follows from Proposition 1. The second line follows by the fact that j was pruned from L_t and thus satisfied the pruning condition. The fourth line follows from the fact that $\sigma(O_{a:c}) \leq \sigma(O_{a:b}) + \sigma(O_{b:c})$ for any a, b, and c. The final line follows from Proposition 1 and the fact that j was assumed to be a final transition point for some member of $C^*_w(O_{1:t'})$. This expression shows that $C(S'_{1:t'}|O_{1:t'}) = C^*(O_{1:t'})$ and thus $S'_{1:t'}$ is in $C^*_w(O_{1:t'})$. Since t and j' are in L_{t+1} and one of them is a final transition point of $S'_{1:t'}$ we have completed the proof. □

Lemma 1. *Given as input an h-constant 2-d grid Potts model and a threshold τ, the problem of deciding whether there is a vertex labeling with cost less than τ is NP-complete.*

Proof. Membership in NP is trivial. Simply use the finite set of vertex labelings as certificates. The cost of each certificate can be evaluated efficiently and then compared to τ.

We show hardness by reduction from the decision problem in Theorem 2 for 2-d grids Potts models. We proceed in two steps. First, given a 2-d grid Potts model, we construct an "equivalent" model that can be derived from an h-constant 2-d grid by removing some vertices and edges. Next, we construct an "equivalent" h-constant 2-d grid model by adding in the missing vertices and edges.

Consider an arbitrary 2-d grid Potts model $P = \langle V, E, D, C_l, C_p \rangle$ and let K be a constant that is greater than the maximum-cost labeling of P, i.e. K is a strict upper bound on cost for P. Such a constant can be efficiently computed by summing the absolute values of all possible labeling and pairwise costs. We now construct a new Potts model $P' = \langle V', E', D, C'_l, C'_p \rangle$ from P, where Figures 4a and b show an example nine vertex model P and the corresponding model P'. For each vertex $v \in V$, the vertex set V' contains six vertices $Q[v] = \{v'_1, \ldots, v'_6\}$, giving that $|V'| = 6 \cdot |V|$. E' contains edges that form a cycle over $Q[v]$ for each $v \in V$, i.e. the edge set $\{(v'_1, v'_2), (v'_2, v'_3), \ldots, (v'_6, v'_1)\}$ is contained in E' and no other edges between vertices in $Q[v]$ are in E'. In addition, for each edge $(v_1, v_2) \in E$ there is a single edge $Q[(v_1, v_2)] = (q_1, q_2)$ in E' where $q_1 \in Q[v_1]$ and $q_2 \in Q[v_2]$. Note that $|E'| = 6 \cdot |V| + |E|$.

For each $v \in V$ we select one member v' of $Q(v)$ and let $C'_l[v', d] = C_l[v, d]$ for all $d \in D$ and for the remaining members of $Q(v)$ we define the labeling cost function so that it always returns zero. As depicted in Figure 4b, the graph corresponding to V' and E' can be embedded in a grid structure where the $Q(v)$ form rectangular cycles (with two vertical edges and four horizontal edges), and the cycles $Q[v_1]$ and $Q[v_2]$ are connected by a single edge iff $(v_1, v_2) \in E$. For each set $Q[v]$, we let $C'_p[e'] = K$ for each of the four horizontal edges e' in the cycle $Q[v]$, and let $C'_p(e') = 2 \cdot K$ for each of the two vertical edges e'. In addition for each $e \in E$ we let $C'_p[Q(e)] = C_p[e]$. As depicted in Figure4b, it is important to note that all such edges $Q[e]$ are vertical edges in our grid embedding—thus all horizontal edges in the embedding have a constant cost of K. However, P' does not form a full 2-d grid, as it is missing many vertices, and thus is not an h-constant 2-d grid model.

For each vertex labeling L of P, let L' be the vertex labeling of P' such that for all $v \in V$ and $v' \in Q(v)$ we have that $L'(v') = L(v)$. We refer to the set of all such labelings of P' as the *feasible labelings* for P'. Note that there is a one-to-one correspondence between feasible labelings of P' and labelings of P. It is easy to verify that for any feasible labeling L' derived from L that the cost of L' under P' is equal to the cost of L under P. Furthermore, we know that for any non-feasible labeling, there must be a $v \in V$ such that at least one pair of neighboring vertices in $Q(v)$ have different labels. Thus, the cost of any non-feasible labeling L' must be at least K. Since K is an upper bound on the cost of any feasible labeling, we know that only feasible labelings need be considered as minimum-cost solutions for P'. This shows that P' is equivalent to P in the sense that P has a labeling with cost less than τ iff P' has a labeling with cost less than τ. Note that we can obtain this same property by constructing a P' with only two vertices in each Q(v) rather than six. The reason we use six is that the "extra space" helps us to convert P' into an h-constant 2-d grid as shown below.

We now show that we can construct an h-constant 2-d grid model P^* that is equivalent to P'. Though P' is embedded in a grid, it is missing many vertices and hence is not a grid. We correct this in two stages. First, we add vertices and edges to the "exterior" of P' to get a new model P'' as shown in Figure 5. The label cost function for each new vertex is always zero (i.e. the label does not matter), the newly added horizontal edges have costs of K, and the vertical edges have costs of zero. Just as for P' it is easy to see that there is a set of feasible labelings of P'' that are in one-to-one correspondence with labelings of P and that all other non-feasible labelings have cost at least K. Thus, P'' has a labeling with cost less than τ iff P does also.

P'' is not yet an h-constant 2-d grid as it has gaps, each one corresponding to two missing vertically aligned vertices. We now show how to "fill in" these gaps to obtain P^*. Figure 6 shows a new model P''_1 that results by filling in a single gap of P'' with a gadget consisting of two vertices t and t', and five edges. The label cost functions for the two new vertices always return zero, the four new horizontal edges have cost K, and the single new vertical edge between t and t' has a cost of $-2 \cdot K$. For any labeling L'' of P'', there is a corresponding set of *extended labelings* of P''_1 that are identical to L'' but consider all possible combinations of labels for t and t'. A straightforward case analysis can be used to verify the following properties:

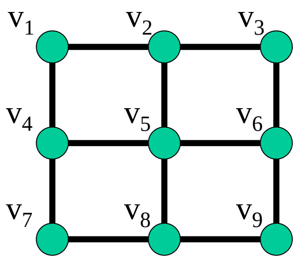
- For any feasible labeling with cost C in P'', there is a corresponding extended labeling with cost C in P''_1.

- For any feasible labeling of P'' with cost C, there is no corresponding extended labeling with cost less than C in P''_1.

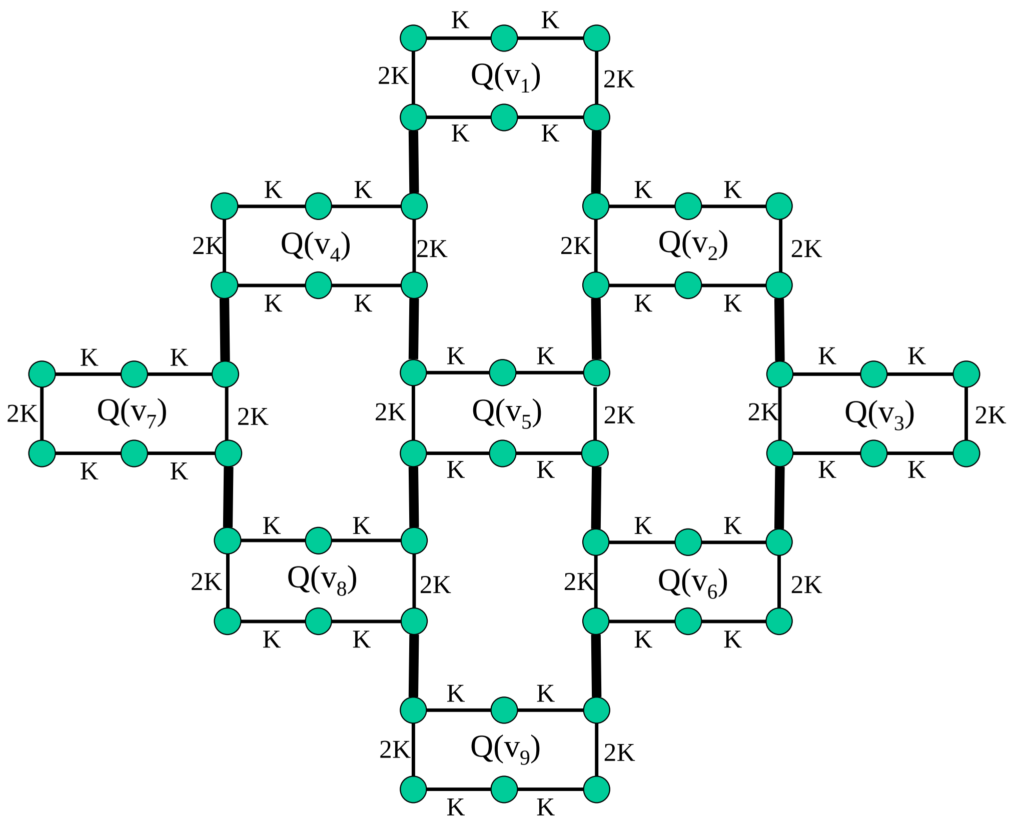
- For any non-feasible labeling of P'', all corresponding extended labelings have a cost of at least K.

Using these properties it is easy to show that P_1'' has a labeling with cost less than τ iff P'' does also. Note that our choice of using six vertices rather than four vertices for each Q(v) in P' allowed space in the grid for our gadget. Now starting with P_1'' we can continue to fill in gaps using our gadget until there are none left. The resulting model P^*, shown in Figure 7, is an h-constant 2-d grid Potts model. Again the above properties allow us to conclude that P^* has a labeling less than τ iff P_1'' does. Hence, as desired, P^* has a labeling with cost less than τ iff P does. Since P was assumed to be an arbitrary 2-d grid Potts model we have described a reduction from 2-d grid Potts models to the restricted case of h-constant 2-d grids. This completes the proof.

$\square$



(a)



(b)

Figure 4: First step of the construction used in the proof of the Lemma. (a) Graphical structure of a nine vertex 2-d grid Potts model. Each edge is associated with a numeric pairwise cost (not shown) and each vertex is associated with a labeling cost function. This model is referred to as model P in the text. (b) Graphical structure of P' which is derived from P as described in the text. Each vertex v_i in P is associated with a cycle of six vertices denoted by $Q(v_i)$. The thick edges between cycles correspond to edges in P.

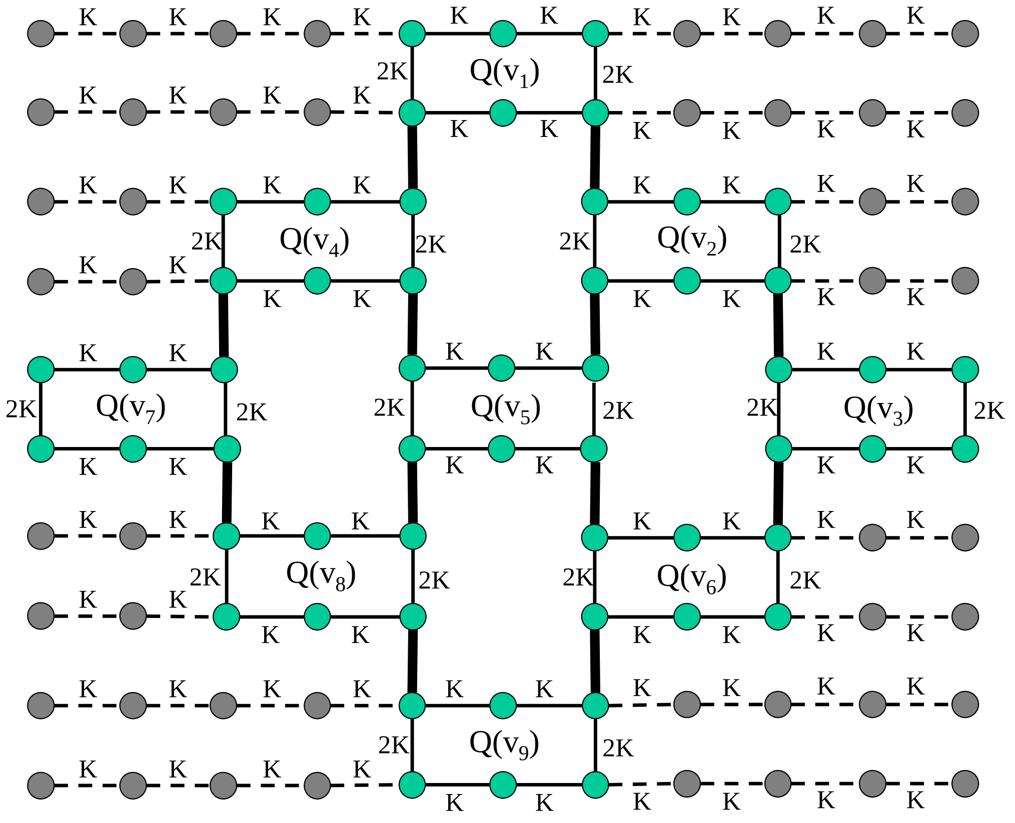


Figure 5: The second step of the construction used in the proof of the Lemma. This is the graph P'' referred to in the text and is constructed by adding exterior vertices and edges to P'. The newly added nodes are connected via dashed edges. The vertical edges connected to these nodes are not shown as they all have a cost of zero. The label cost function for these newly added nodes always returns zero. Note that this graph is still not a complete 2-d grid because of the four interior gaps, where each gap corresponds to two missing vertices.

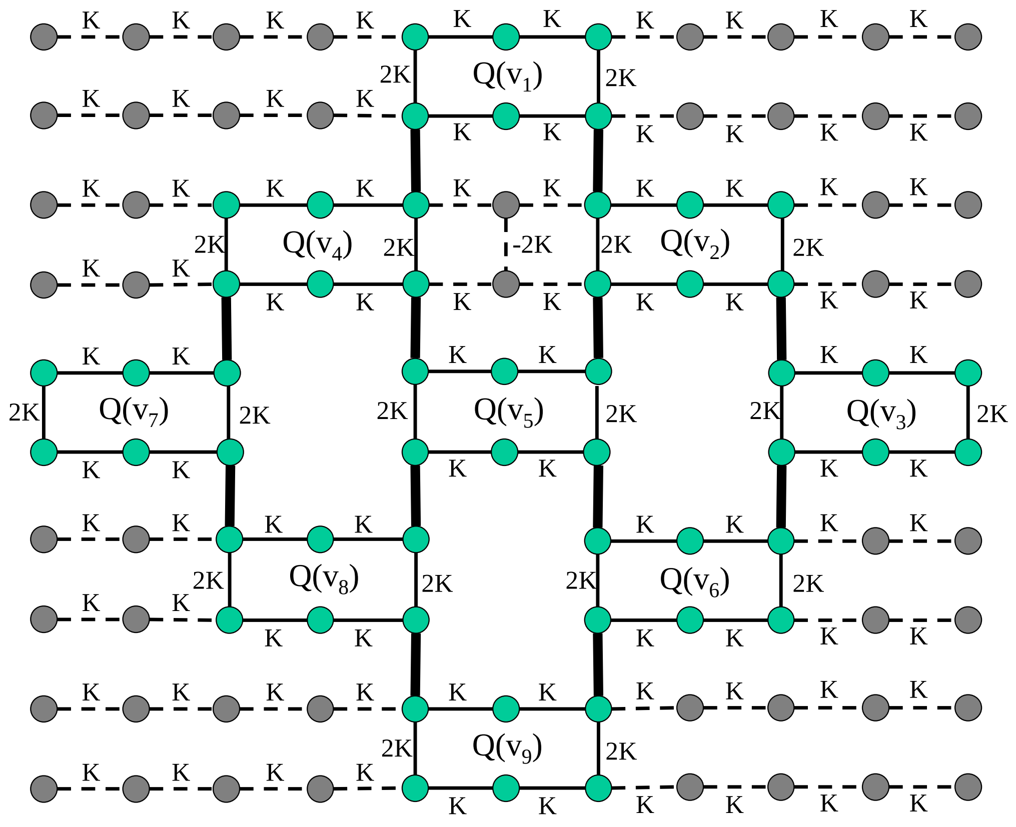


Figure 6: The third step of the construction used in the proof of the Lemma. This is the graph P_1'' referred to in the text and is the result of filling in one of the interior gaps in P''. The two newly added vertices t and t' have a pairwise cost of $-2 \cdot K$ between them, which allows us to prove the necessary properties. Note that all horizontal edges in the graph have a cost of K.

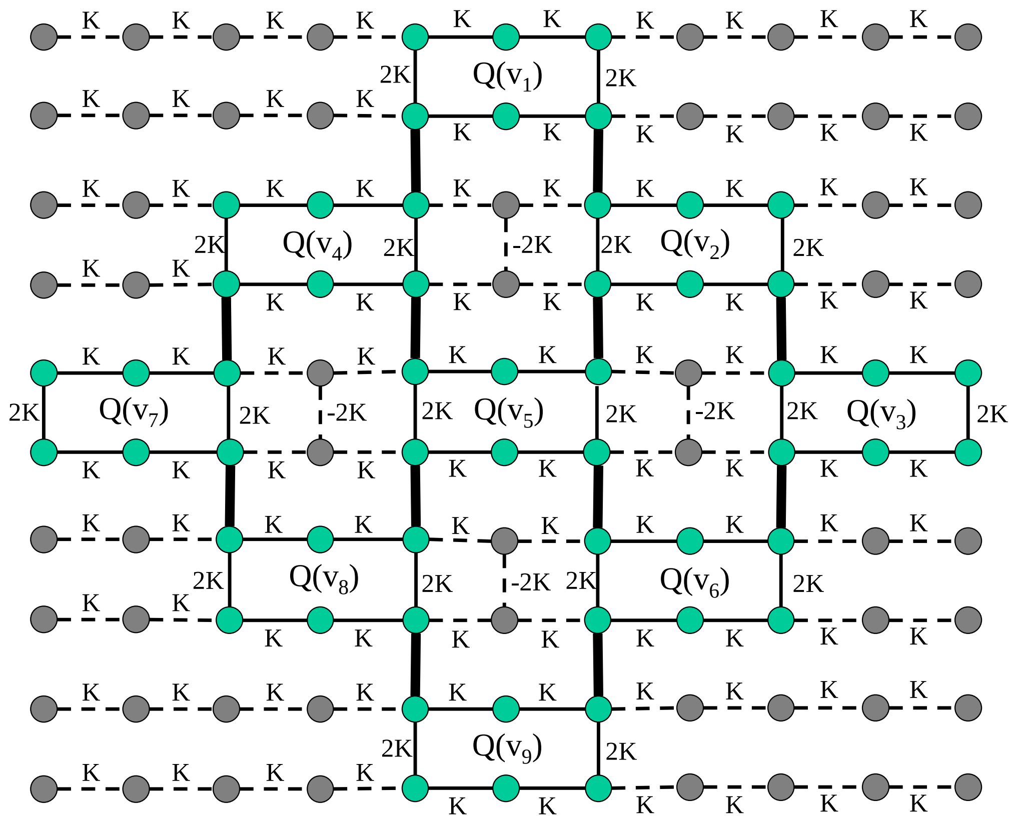


Figure 7: The final step of the construction used in the proof of the Lemma. This is the graph P^* referred to in the text and is the result of filling in the remaining gaps in P_1''. Any vertical edges not shown have a cost of zero. Note that all of the horizontal edges have a cost of K. Thus this model is an h-constant 2-d grid. The text argues that there is a labeling for P^* with cost less than τ iff there is a labeling for P with cost less than τ.