UMINF 10.06
Weighted Unranked Tree Automata as a Framework for Plan Recognition

Johanna Högberg
Dept. of Computing Science
Umeå University
Sweden.
johanna@cs.umu.se

Lisa Kaati[*]
FOI
Swedish Defence Research Agency
Sweden.
lisa.kaati@foi.se

Abstract – *As the amount of information accessible to military intelligence continues to surge, manual surveillance becomes more and more inefficient. To process the information stream efficiently, automatic systems for threat detection are called for. These systems must be sufficiently robust to process incomplete or noisy data, and capable of dealing with uncertainties and probabilities. For safety reasons and accountability, it is imperative that the surveillance systems are specified in a formal framework that allows for rigorous mathematical verification. To this end, we demonstrate how the unobstructed keyhole plan recognition problem can be modelled within the framework of weighted unranked tree automata, and outline a software system for recognition of hostile behavior.*

Keywords: plan recognition, activity recognition, weighted unranked tree automata, impact assessment.

1 Introduction

Plan recognition is the task of inferring an agent's plans and goals, based on observations of the agent's actions, or the effects of those actions [10, 42]. Different fields refer to plan recognition as goal-, intent-, behavior -, or activity recognition. The latter term is common within medicine, where it has important applications for, e.g., the rehabilitation of patients suffering from dementia. Presently, extensive efforts are directed towards the development of systems that can aid elderly patients in their everyday routines, thus empowering them and making them more self-reliant.

In this paper, we view plan recognition (PR) as a means for automatic threat detection. Military intelligence typically receives information from a wide range of sources such as field operatives, sensors, political delegators, public news flows, surveillance, and administrative agencies. To derive predictions about critical events from such heterogenous data is as important as it is difficult. For this reason, software systems have been developed to support human analysts in their task. One example is Impactorium, a collection of software tools for information fusion developed by the Swedish Defence Research Agency (FOI) [20]. Our objective is to extend Impactorium with a component for PR-based threat detection. To guarantee correctness and performance, we first model PR in a formal framework, and then translate the model into a software architecture.

In general, plan recognition relies on a *plan library* that contains abstract descriptions of plans that can be executed by the observed agent. The plan library is usually maintained by a human analyst and consists of a set of top-level plans that are decomposed hierarchically. At the highest level we find the ultimate goal that motivates a plan, and at the lowest level, observations that indicate that the plan is active. The remaining levels contain intermediate goals that provide structure and simplify understanding. [3]

The *plan recognition problem* consists of deducing from a set of observations which plans in a given library are being realised. Plan recognition problems can be classified as either *intended*, or *keyhole*, or *obstructed* [27]. In intended plan recognition it is assumed that agents are deliberately structuring their activities to make their intentions clear, whereas in keyhole plan recognition this is not the case. In obstructed plan recognition, agents may even try to perform erroneous plans [48] to confuse potential observers.

The abstract machinery used to solve this problem is called a *plan recognizer*. As we will see in the section on related work, the literature contains a variety of plan recognizers and theoretical frameworks geared for this purpose. Two noticeable paradigms are *symbolic plan recognition* [27, 3] and *probabilistic plan recognition* [30, 37]. The former focuses on the consistency of the plans in a plan library, whereas the latter consists in computing the likelihood of each plan when the observations are given as priors, and then reporting those plans that scored the highest to the user. Symbolic and probabilistic plan recognition can be combined with decision-theoretic plan recognition [4] which is based on the idea of maximizing a known utility function.

In this paper, we model the unobstructed keyhole plan recognition problem within the framework of weighted un-

[*]This work was supported by the FOI research project *Tools for information management and analysis*, which is funded by the R&D programme of the Swedish Armed Forces.

ranked tree automata (wuta). Tree automata capture the hierarchical structure nicely and brings a wealth of theoretical results and algorithms to the table [7, 13, 14, 36, 46, 47, 40, 41]. In unranked as opposed to ranked tree automata, a node in a tree (which is now analogous to an intermediate goal in a plan), is allowed to have an unbounded number of children. This makes it easier to express relations such as *if we have seen two or more swallows (up to an unbounded number), then it is probably summer*. By using weighted devices, we can prioritize the plans in the library by impact factor, or by likelihood of activation with respect to a given sequence of observations. Our approach to the plan recognition problem thus adheres to the probabilistic paradigm.

1.1 Related work

Let us now review previous efforts in the field. In [28], plans and plan decompositions are represented as graphs with top-level actions as root nodes and expansions of these actions into unordered sets of child nodes. The method described in [28] assumes that an agent only aims at accomplishing one goal at a time and the method does not take into consideration the likelihood of the possible plans.

In [21], an abductive probabilistic algorithm for task tracking/intent inference called PHATT is presented. PHATT works in a a three-stage process. First it computes the complete and covering set of possible explanations; second it computes the probability of each of the explanations. Third it computes the conditional probability of the given goal on the basis of the probability of the explanations.

In [3], the focus is on matching observations with plans efficiently. The authors use a hierarchical representation of the plan library and present a method for efficiently matching multi-feature observations to the library. To speed up the process of matching observations to plans in the plan library they use a Feature Decision Tree (FDT) that maps observations to matching nodes in the plan library. Ideally, each leaf node of the FDT points to only one plan that matches the conjunctive set of observations made.

Instead of matching observations to a given plan, a situation model is extracted from estimated track data in [35]. The situation model is used to obtain situation assessment and it is valid for a finite time. During this time, the state of the situation can be evaluated. Another grammar-based approach towards situation and threat analysis is described in [34]. In this paper situation trees are extracted from a *sequence set combinatory categorial grammar*. A situation tree is a formalism that can be used to represent the semantics of a battle. A node in a situation tree contains information about its duration, type and role.

An alternative approach to plan recognition is described in [39]. Instead of using a plan library they use a *domain* and the set of goals that explain a sequence of observations given a domain theory is generated. The (unweighted) set of all possible goals is generated.

2 Preliminaries

Before we continue, let us recall the theoretical fundament of weighted unranked tree automata. A complete presentation is of course not possible within the scope of this paper, but the reader will find a nice introduction to tree automata in [12] and a survey of weighted automata in [17].

Sets, numbers, and relations. To cover the syntax and semantics of weighted devices, we need a bit of algebra. The set of all natural numbers including 0 is denoted by $\mathbb{N}$. The subset $\{1, 2, \ldots, n\}$ of $\mathbb{N}$ is abbreviated by $[n]$. Note that $[0] = \emptyset$. The *permutations* of $[n]$, where $n \in \mathbb{N}$, is the set $\mathcal{P}([n])$ of bijections from $[n]$ onto $[n]$.

Let V be a vector space over the carrier set K with dimensions indexed by the set S. For every $s \in S$, the vector $e_s : S \to K$ is such that $e_s(s) = 1$, and $e_s(s') = 0$ for every $s' \neq s$ in S. In other words, e_s is a unit vector with a single non-zero component; a one at the position indexed by s.

An *alphabet* Σ is a finite nonempty set of symbols. We denote the empty string by ε and the length of a string $w \in \Sigma^*$ by $|w|$. Let $w = f_1 \cdots f_k$ with $f_i \in \Sigma$ for every $i \in [k]$; the *label* at position $i \in [k]$ of w is $w(i) = f_i$. The set of *shuffles* of strings $u, v \in \Sigma^*$, denoted $u \odot v$, is defined inductively: $u \odot \varepsilon = \varepsilon \odot u = \{u\}$, for every $u \in \Sigma^*$, and

$$\sigma u \odot \gamma v = \{\sigma w \mid w \in u \odot \gamma v\} \cup \{\gamma w \mid w \in \sigma u \odot v\} \ ,$$

for every $\sigma, \gamma \in \Sigma$, and $u, v \in \Sigma^*$.

Algebraic structures. Let K be a nonempty set, and let $\cdot$ be an associative binary operation on K. If K contains an element 1 such that $1 \cdot \kappa = \kappa = \kappa \cdot 1$ for every $\kappa \in K$, then $(K, \cdot)$ is a *monoid* with *identity* 1. A monoid $(K, \cdot)$ is *commutative* if the equation $\kappa_1 \cdot \kappa_2 = \kappa_2 \cdot \kappa_1$ holds for every $\kappa_1, \kappa_2 \in K$. We henceforth adopt the convention of identifying an algebraic structure with its carrier set.

By combining a pair of monoids that share the same carrier set and fulfill a number of additional criteria, we obtain the semirings. Representatives of this algebraic structure are often used as domains of weighted devices. The theory of semirings is presented in [23].

A *commutative semiring* is a nonempty set K on which a binary addition $+$ and a binary multiplication $\cdot$ have been defined, such that the following conditions are satisfied:

- $(K, +)$ and $(K, \cdot)$ are commutative monoids with identities 0 and 1, respectively;
- the operation $\cdot$ distributes over $+$ from both sides; and
- 0 is absorbing (i.e., $0 \cdot \kappa = 0 = \kappa \cdot 0$ for every $\kappa \in K$).

A semiring is *zero-sum free* if $\kappa_1, \kappa_2 \in K \setminus \{0\}$ implies that $\kappa_1 + \kappa_2 \in K \setminus \{0\}$. The *support* of a function $f : S \to K$, where K is a semiring, is the subset of S on which f is nonzero, i.e., $support(f) = \{s \in S \mid f(s) \neq 0\}$. Henceforth, we only consider commutative zero-sum free semirings, as commutativity allows us to reorder factors freely, and zero-sum freeness simplifies the problem.

Weighted string automata. In the unweighted or Boolean setting, an unranked tree automaton classifies an input tree (labelled with symbols in some alphabet Σ) as inside or outside a target language. In the weighted setting, it associates a semiring element with the tree. In either case, the automaton consists of component string automata, one for each symbol in Σ. These string automata are a bit special in that rather than just answering yes or no in response to an input string, they produce an *output state*, taken from a set P. If the automaton is nondeterministic, it may produce a set of output states. The tree automata in this contribution process their input trees bottom-up, treating one node at the time. When the computation reaches a node v labelled σ, below which there are n subtrees that have already been mapped to output states $q_1, \ldots, q_n$, we execute the string automaton associated with σ on the string $q_1 \cdots q_n$, and associate the output state produced by this run with v. The computation continues in this fashion, until the entire input tree has been treated.

To properly define automata over unranked trees, we thus need to define automata with P-output over strings. Already at this point we include weights in the definition.

A *weighted string automaton with P-output* (abbreviated P-wsa) [43, 18] is a tuple $A = (Q, \Sigma, K, \lambda, \mu, \nu)$ where

- Q is an alphabet of *states*;
- Σ is an alphabet of *input symbols*;
- K is a semiring;
- $\mu : \Sigma \to K^{Q \times Q}$ assigns a *transition weight matrix* to each symbol;
- $\lambda \in K^Q$ is an *initial weight vector*; and
- $\nu \in K^{Q \times P}$ is a *final weight vector*.

The *size* of the wsa A, denoted by $size(A)$, is $n + m$, where n is the number of states and m is the total number of non-zero entries in the transitions weight matrices. The mapping $\mu : \Sigma \to K^{Q \times Q}$ uniquely extends to a monoid homomorphism $\overline{\mu}$ from $(\Sigma^*, \cdot)$ to $(K^{Q \times Q}, \cdot)$. The formal series $\mathcal{S}(A) \colon \Sigma^* \to K^P$ *recognised by A* is defined for every $w \in \Sigma^*$ by $\mathcal{S}(A)(w) = \lambda \cdot \overline{\mu}(w) \cdot \nu$.

Unranked trees and ranked trees. Let Σ be an alphabet. The set U_Σ of *(unranked) trees over* Σ is the smallest subset of $(\Sigma \cup \{[,]\} \cup \{,\})^*$ such that for every $\sigma \in \Sigma$, $k \in \mathbb{N}$, and $t_1, \ldots, t_k \in \mathrm{U}_\Sigma$ also $\sigma[t_1, \ldots, t_k] \in \mathrm{U}_\Sigma$. To improve readability, we henceforth identify $\sigma[]$ with σ. An *(unranked) tree language* (over Σ) is a subset of U_Σ.

When we later use trees to represent plans, the nodes of the tree (or *positions*) will come to represent intermediate goals and observable activities. The set of positions in the tree t, denoted by $pos(t)$, is inductively defined for every $t = \sigma[t_1, \ldots, t_k] \in \mathrm{U}_\Sigma$ by $pos(t) = \{\varepsilon\} \cup \{iw \mid 1 \leq i \leq k \text{ and } w \in pos(t_i)\}$. The *leaves* of t is the set of positions $\{v \in pos(t) \mid \nexists u \in \mathbb{N}^* s.t.\ vu \in pos(t)\}$.

The *yield* of a tree is the string of symbols that labels its leaves, when read from left to right. More formally, the yield of a tree t $yield(t)$ is inductively defined for every $t = \sigma[t_1, \ldots, t_k] \in \mathrm{U}_\Sigma$ by $yield(t) = \sigma$ when $k = 0$, and $yield(t) = yield(t_1) \cdots yield(t_k)$ otherwise. The yield of a tree language $\mathcal{L}$ is $\{yield(t) \mid t \in \mathcal{L}\}$.

Let $t = \sigma[t_1, \ldots, t_k] \in \mathrm{U}_\Sigma$ and $w \in pos(t)$. The *rank of t at w* and the *label of t at w* are denoted by $rank_t(w)$ and $t(w)$, respectively. They are defined as follows: $rank_t(\varepsilon) = k$ and $t(\varepsilon) = \sigma$; and $rank_t(iv) = rank_{t_i}(v)$ and $t(iv) = t_i(v)$, for every $i \in [k]$ and $v \in pos(t_i)$.

A *ranked alphabet* is an alphabet Σ together with a mapping $rk : \Sigma \to \mathbb{N}$. The set T_Σ of *ranked trees* over Σ is the subset of U_Σ that is given by

$$\mathrm{T}_\Sigma = \{t \in \mathrm{U}_\Sigma \mid \forall w \in pos(t) : rank_t(w) = rk(t(w))\} \ .$$

A mapping $\mathcal{S} \colon \mathrm{U} \to K$, where $\mathrm{U} \subseteq \mathrm{U}_\Sigma$, is called an *(unranked) tree series*. When $\mathrm{U} \subseteq \mathrm{T}_\Sigma$ we also say that $\mathcal{S}$ is a *ranked tree series*.

Weighted (unranked) tree automata. A *weighted unranked tree automaton* (wuta, when abbreviated) is a system $M = (Q, \Sigma, K, (A_f)_{f \in \Sigma})$ where

- Q is a set of *states*;
- Σ is an alphabet of *input symbols*;
- K is a semiring; and
- $A_\sigma = (Q_\sigma, Q, K, \lambda_\sigma, \mu_\sigma, \nu_\sigma)$ is a wsa with Q-output for every $\sigma \in \Sigma$.

Wuta are sometimes defined to include a *root weight vector*, which we omit as it does not influence later constructions.

For the remainder of this section, let M be the wuta $(Q, \Sigma, K, (A_\sigma)_{\sigma \in \Sigma})$. The size of M is $size(M) = |Q| + \sum_{\sigma \in \Sigma} size(A_\sigma)$. For the sake of simplicity, we henceforth assume, without loss of generality, that $Q_\sigma \cap Q_{\sigma'} = \emptyset$ and $Q_\sigma \cap Q = \emptyset$, for every $\sigma, \sigma' \in \Sigma$ such that $\sigma \neq \sigma'$.

We continue with the definition of the semantics. Let $t \in \mathrm{U}_\Sigma$. A *run* r of M on t is a mapping $r \colon pos(t) \to Q$, and we denote by $runs_M(t)$ the set of all runs of M on t. Furthermore, the *weight* of r is

$$weight_M(r) = \prod_{\substack{w \in pos(t) \\ k = rank_t(w)}} (\mathcal{S}(A_{t(w)}), r(w1) \cdots r(wk))_{r(w)} \ .$$

The *unranked tree series* recognised by M is defined for every $t \in \mathrm{U}_\Sigma$ by

$$\mathcal{S}(M)(t) = \sum_{r \in runs_M(t)} weight_M(r) \ .$$

Using the yield mapping, we can associate every wuta M over the alphabet Σ and semiring K with a string series $\hat{\mathcal{S}}(M) : \Sigma^* \to K$. The series is defined as follows:

$$\hat{\mathcal{S}}(M)(w) = \sum_{\substack{t \in \mathrm{U}_\Sigma \\ yield(t) = w}} \mathcal{S}(M)(t) \ .$$

Let us conclude with an informal discussion of *weighted tree automata* and *weighted context-free grammars*. A weighted tree automaton (wta) is a wuta in which the support of the string series computed by every component tree automaton is finite. A weighted context-free grammar (wcfg) is a context-free grammar G in which every rewrite rule has an associated weight. A wcfg with alphabet Σ and weights in the semiring K defines a mapping $\hat{\mathcal{S}}(G) : \Sigma^* \to K$. The value of $\hat{\mathcal{S}}(G)(w)$ is the sum of the weight of every derivation of w in G; the weight of a derivation d is the product of the weights of the rewrite rules used in d. The two devices are strongly related: for every wuta M, there is a wcfg G such that $\hat{\mathcal{S}}(M) = \hat{\mathcal{S}}(G)$, and vice versa.

3 Choice of framework

As mentioned in the introduction, we wish to construct a software system for automatic threat detection, thus implementing ideas and techniques from the field of probabilistic plan recognition. To prove correctness and estimate the computational efficiency of the system, it is necessary to give precise definitions of data types and algorithms. In this section, we argue why weighted unranked tree automata are an appropriate formal framework for plan recognition.

Capable of modelling timed events. Plans, and the observable activities that they suggest, are often expected to be realised in a certain order. To keep the hypothesis space small, we want to describe these time-related dependencies using the mechanics of our formalism. Although Bayesian networks provide a good representation for the probabilistic interdependencies of events [38, 2, 32], they do not capture sequential behavior well. Instead, when events are ordered along a timeline, *Markov models* (mm) are a more common choice [9, 8]. However, Markov models do not seem appropriate to represent plans as they fail to capture nested dependencies: the support of the probability distribution computed by a Markov model is a regular language, but the surface language (or yield) of a hierarchical task network is context-free, so there is a clear mismatch. One solution is to use the more expressive *recursive* Markov model (rmm), i.e., a set of component Markov models that can invoke each other in a recursive manner. There are striking resemblances between recursive Markov models and wuta, but we prefer the latter. The relations between rmm and wuta will be further investigated in future work.

Well-researched theory. Unranked tree automata are a generalisation of ranked tree automata, a formal device that has a particularly extensive theory [7, 13, 14, 36, 46, 47, 40, 41]. As shown in [29], unranked trees can be encoded as ranked trees, using a *first-child-next-sibling* encoding. This result makes it possible to reduce many problems from the unranked to the ranked setting. For example, the parse problem for wuta can be reduced to that for wta in quadratic time [11].

Extensive algorithmic toolbox. Both ranked and unranked tree automata were originally motivated by practical applications. Ranked tree automata were designed to model linguistic theories and are now a standard component in natural language processing (nlp). Unranked tree automata were introduced to provide mathematical rigour to xml, a language for structuring data, and this made them useful in database theory. In light of this, it is not surprising that efficient algorithms have been developed for a wide range of automata-related tasks, including learning [16], minimising [7, 33, 24], searching [44], querying [11, 31], and analysing [19] tree automata. As pointed out in [22], it is desirable that these algorithms should be transfered to, rather than reinvented in, the field of plan recognition.

Formal verification. Regular model checking is a set of techniques for analysis of transition systems [6, 26]. In automata theory, these ideas were initially applied to string automata, and later casted to tree automata [1, 15]. In automata theory, model checking can be used to determine properties such as transitive closure and termination. A property of immediate practical importance is that of reachability. Given a set of goals, it is possible to calculate the set of initial observations that are needed to reach the goals using backward reachability. Similarly, forward reachability can be used to decide if it is possible to reach a given goal, starting from a set of initial observations. Reachability can thus be used to verify that a proposed set of countermeasures will prevent an actual threat from being realized.

4 Modelling plans

Let us now demonstrate how wuta can be used to model the plan recognition problem. We begin by defining what constitutes a well-formed plan, and then explain what is required to recognise a plan in a sequence of observations.

Syntax. For the remainder of this section, let Λ denote a fixed but arbitrary alphabet of *indicators*. A plan is an unranked tree in which indicators label leaves, and *plan operation symbols* label internal nodes. The indicators correspond to (abstract classes of) observable activities, and the internal nodes represent intermediate goals. Each internal node v has an associated weight that reflects its importance or likelihood in the context of the overall plan. More formally:

Definition 4.1 (Plan)**.** *Let* $\Theta = \{\exists, \forall, \overline{\forall}\}$ *be the set of* plan operator symbols. *The set* $\mathrm{P}_{\Lambda,K}$ *of* weighted plans *over the alphabet* Λ *and the semiring* K *is the smallest subset of* $\mathrm{U}_{(\Theta\cup\Lambda)\times K}$ *such that* $(\Lambda \times (K \setminus \{0\})) \subseteq \mathrm{P}_{\Lambda,K}$*, and if* $(\sigma, \kappa) \in (\Theta \times (K \setminus \{0\}))$ *and* $p_1, \dots, p_n \in \mathrm{P}_{\Lambda,K}$*,* $n \in \mathbb{N}$*, then* $(\sigma, k)[p_1, \dots, p_n] \in \mathrm{P}_{\Lambda,K}$*.*

In practice, it is often useful annotate the subtrees of a plan with descriptive names that explain what intermediate goals they are to accomplish. This has for example been done in Figure 1 to aid understanding. Since the names do not affect the formal semantics of a plan, they are omitted in the mathematical part of the presentation.

Semantics. A plan that consists in a single pair $(\sigma, \kappa) \in \Lambda \times K$ is realised by the observation σ. A more complex plan can typically be realised in a number of different ways, depending on the operator symbols that label its internal nodes. If a plan p is of the form $(\exists, \kappa)[p_1, \dots, p_n]$, then p is realised if at least one of $p_1, \dots, p_n$ is realised. If $p = (\forall, \kappa)[p_1, \dots, p_n]$, then p is realised if all of $p_1, \dots, p_n$ are realised, in any order. Finally, if $p = (\overline{\forall}, \kappa)[p_1, \dots, p_n]$, then p is realised if all of $p_1, \dots, p_n$ are realised, in that particular order.

To simplify the formal definition of the semantics, we need the following technicality: For every number $i \in \mathbb{N}$ and tree $t = v[t_1, \dots, t_n] \in \mathrm{U}_{\mathbb{N}^*}$, we denote by $i \cdot t$ the tree $iv[i \cdot t_1, \dots, i \cdot t_n]$.

Definition 4.2 (Realisations)**.** *The set of* realisations *of a plan* $p = (\sigma, \kappa)[p_1, \dots, p_n] \in P_{\Lambda,K}$ *(denoted* $\mathcal{L}(p)$*) is the*

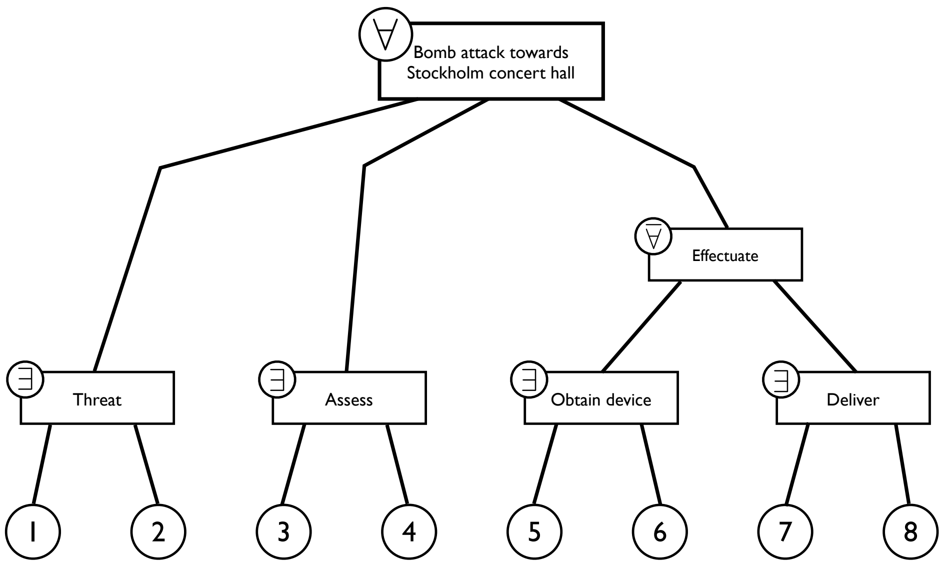

Figure 1: A plan describing how a hostile group may proceed when targeting the Stockholm concert hall. The symbols $\exists, \forall$ and $\overline{\forall}$ that can label an internal node n indicate when the plan associated with n is active, in terms of the activation of the subplans below n. To save space and improve readability, we give the labels of the leaves as text in Example 1.

subset of $\mathrm{U}_{pos(p)}$*, defined by the following table:*

σ	$\mathcal{L}(p)$
$\in \Lambda$	$\{\varepsilon\}$
$\exists$	$\{\varepsilon[i \cdot t] \mid i \in [n]$ *and* $t \in \mathcal{L}(p_i)\}$
$\overline{\forall}$	$\{\varepsilon[1 \cdot t_1, \ldots, n \cdot t_n] \mid t_i \in \mathcal{L}(p_i), i \in [n]\}$
$\forall$	$\{\varepsilon[1 \cdot t_1, \ldots, n \cdot t_n] \mid f \in \mathcal{P}([n])$ *and* $t_i \in \mathcal{L}(p_{f(i)})\}$

To illustrate Definitions 4.1 and 4.2, consider Example 1.

Example 1. *This year, Winston Smith, the author of several controversial books about Ideas, was awarded the Nobel prize in literature. On December 10, the Nobel Prize Award Ceremony and the Nobel Banquet in Stockholm take place and the author arrives to Stockholm the same day as the ceremony. The Swedish government has received several threats directed towards Mr Smith and they are worried that an attack may be directed towards the award ceremony. The ceremony takes place in Stockholm concert hall which is located in the heart of the city, so it is extremely important to detect and prevent any kind of explosive attack.*

To help surveillance, a human analyst draws up a set of threat models, i.e., plans. Each plan describes one course of action that an adversary group may take in order to attack the concert hall. Given the above scenario, the analyst's work could for instance include (a more elaborated version of) the plan in Figure 1. For the sake of readability, we give the observable activities as text. The leaves of the plan in Figure 1 are thus as follows.

1. *Increased activity in activist forums.*
2. *Video clips containing explicit threats appear online.*
3. *Blueprints of the concert hall are solicited from the municipal office.*
4. *Inquiries are made regarding the accommodation of Mr Smith during the festivities.*
5. *Theft of explosive materials from construction sites.*
6. *Purchases of large quantities of synthetical fertilizers.*
7. *Last-minute replacements of personnel.*
8. *Parked cars by the concert hall entrance.*

Sequences of observations that correspond to realisations of the plan include e.g. $(2, 5, 6, 8, 1, 3)$ *and* $(1, 3, 6, 5, 8, 7)$*, but not* $(2, 5, 6, 8, 1)$ *since it does not realise the intermediate goal of assessing the venue, nor* $(1, 3, 8, 6)$ *since the explosive device must be obtained before it can be delivered.*

To estimate the impact of a plan being realised in a certain way, we multiply the weights of the intermediate goals accomplished during this particular realisation.

Definition 4.3 (Computed tree series, weight)**.** *The tree series* $\mathcal{S}(p) : \mathcal{L}(p) \to K$ *computed by* $p \in \mathrm{P}_\Lambda$ *is given by*

$$\mathcal{S}(p)(t) = \prod_{v \in pos(t)} wgt(p(t(v))) \enspace ,$$

where $wgt : \Sigma \times K \to K$ *is defined by* $wgt((\sigma, \kappa)) = \kappa$*, for every* $(\sigma, \kappa) \in \Sigma \times K$*. The tree series* $\mathcal{S}(p)$ *is extended from* $\mathcal{L}(p)$ *to* $\mathrm{U}_{\mathbb{N}^*}$ *by letting* $\mathcal{S}(p)(t) = 0$ *for every* $t \notin \mathcal{L}(p)$*. The* weight *of a tree* $t \in \mathrm{U}_{\mathbb{N}^*}$ *with respect to* $p \in \mathrm{P}_\Lambda$ *is* $\mathcal{S}(p)(t)$*.*

Computation. The problem can now be restated as:

Definition 4.4 (The plan recognition problem)**.** *Given a set of plans* $P \subseteq \mathrm{P}_\Lambda$ *with weights in* K *and a string of observations* $w \in \Lambda^*$*, find*

$$\operatorname*{arg\,max}_{t \in \mathcal{L}(P) \cap yield^{-1}(w)} (\mathcal{S}(p)(t)) \enspace ,$$

where $\mathcal{L}(P) = \cup_{p \in P} \mathcal{L}(p)$*.*

Thus expressed, the plan recognition problem consists in finding the realisation of a plan in P that best explains w. To solve an instance (P, w) of the problem, we translate every $p \in P$ into a wuta M_p, such that $\mathcal{S}(p) = \mathcal{S}(M_p)$. The construction of M_p from p is given below. Due to the closure properties of wuta, we can derive a wuta M with $\mathcal{S}(M)(t) = \sum_{p \in P} \mathcal{S}(M_p)(t)$ in linear time. The wuta M can then be translated into a wcfg G such that $\hat{\mathcal{S}}(M) = \hat{\mathcal{S}}(G)$ in quadratic time. Finally, the k realisations in $\mathcal{L}(P) \cap \mathit{yield}^{-1}(w)$ that have the greatest weight with respect to G can be computed in time $O(mn \log k)$, where m and n are the number of production rules and the number of nonterminals, respectively, of G [25].

Definition 4.5 (Automata representation)**.** *The* wuta *representation M_p of a plan $p = (\sigma, \kappa)[p_1, \ldots, p_n]$ over the semiring K is the wuta $M_p = (Q, \Sigma, K, (A_v)_{v \in pos(p)})$, where $Q = \Sigma = pos(p)$, and A_{iv} is obtained from A_v in M_{p_i} by replacing every occurrence of $u \in pos(p_i)$ in the definition of A_v with iu. Finally, the wsa A_ε depends on (σ, κ) as follows.*

If $\sigma \in \Lambda$ (so $n = 0$) then $A_\varepsilon = (\{q_0\}, Q, K, e_{q_0}, \mu, \nu)$, where μ_q is a zero matrix of dimensions 1×1 for every $q \in Q$; and $\nu(q_0, \varepsilon) = \kappa$.

If $\sigma = \exists$ then $A_\varepsilon = (\{q_0, \ldots, q_n\}, Q, K, e_{q_0}, \mu, \nu)$, where $\mu_j(q_i, q_j) = 1$ for every $i, j \in \{0, \ldots, n\}$ s.t. $i < j$, and 0 otherwise; and $\nu(q_i, \varepsilon) = \kappa$ for every $i \in [n]$ and 0 otherwise.

If $\sigma = \overrightarrow{\forall}$ then $A_\varepsilon = (\{q_0, \ldots, q_n\}, Q, K, e_{q_0}, \mu, \nu)$, where $\mu_i(q_{i-1}, q_i) = 1$ for every $i \in [n]$ and 0 otherwise; and $\nu(q_n, \varepsilon) = \kappa$ and 0 otherwise.

If $\sigma = \forall$ then $A_\varepsilon = (P, Q, K, \lambda, \mu, \nu)$, where

- *$P = \{q_{(f,i)} \mid f \in \mathcal{P}([n]), i \in \{0\} \cup [n]\}$;*
- *$\lambda(q_{(f,0)}) = 1$ for every $f \in \mathcal{P}([n])$ and 0 otherwise;*
- *$\mu_{f(i)}(q_{(f,i-1)}, q_{(f,i)}) = 1$ for every $i \in [n], f \in \mathcal{P}([n])$ and 0 otherwise; and*
- *$\nu(q_{(f,n)}, \varepsilon) = \kappa$ for every $f \in \mathcal{P}([n])$ and 0 otherwise.*

By the construction in Definition 4.5, the wuta M_p computes the semantics of p.

Proposition 4.6. *$\mathcal{S}(M_p) = \mathcal{S}(p)$ for every $p \in \mathrm{P}_{\Lambda,K}$.*

5 Implementation

The formal model presented in Section 4 can be used to draft a software system for automatic threat detection. Figure 2 shows an outline. The core components of the system are an xml-based modelling tool, a plan library, a chart parser, and an information engine. Plans are created and edited by a human analyst using the modelling tool, and stored in the plan library, which is typically a relational database. For each plan, the analyst specifies a pair of weights that express the impact of the plan being active, and an a priori likelihood that the plan will be recognised. Weights can also be associated with intermediate goals to express their importance for the fulfillment the overall plan. To perform threat detection, the plan library is compiled into a wuta, which is then turned into a wta/wcfg. Later additions and changes to the library only require incremental updates of the wcfg to guarantee correctness, although a complete re-compilation will often yield a smaller wcfg. The input to the system is a stream of observations provided by the information engine. Each observation is equipped with a time stamp and a weight reflecting its reliability.

The chart parser searches the input stream for realisations of the plans in the plan library. For efficiency reasons, it maintains a table of partial parses that can be reused throughout the computation [45]. Since the plan library is expected to be of considerate size, it is likely that many realisations can be identified simultaneously in the input stream. To help the operator prioritise between them, the system computes the k realisations that are most relevant with respect to the input stream w. The relevance of a realisation t with yield w of the plan p is the product of p's impact- and probability-weights, as assigned by the analyst, and $\mathcal{S}(p)(t)$.

In practice, the input stream is likely to contain incomplete as well as superflous information, so the chart parser must be revised to reflect this. For instance, since several plans can be active at once it would be useful to parse for shuffles of words. That is, given plans $p_1, \ldots, p_k$ and an observation stream $w \in \Lambda^*$, we search for $u_i \in \mathit{support}(\hat{\mathcal{S}}(p_i))$, $i \in [k]$, and a $v \in \Lambda^*$ such that $w \in u_1 \odot \cdots \odot u_k \odot v$. As shown in [5], this problem is NP-complete unless k is bounded, or further restrictions are added. One possibility is to require that no distinct plans predict overlapping sets of observations. This is severe, but when the condition is met, we avoid the surge in complexity.

As the input to the system is a stream of observations rather than a string, old observations will eventually be considered irrelevant, and new observations will be appended to the stream. In the parsing algorithm, this is handled by deleting obsolete sections of the table when observations timeout, and adding new sections when additional observations are made (see Figure 3). Thanks to the dynamic nature of chart parsers, the contents of new sections are fairly cheap to compute based on previous results.

6 Future work

In Section 4, we interpreted plan recognition in the framework of wuta theory. A natural continuation of this effort is to compile an inventory over algorithms for wuta and wta in xml and nlp, and evaluate which of these are meaningful to translate to the field of plan recognition. For instance, algorithms for learning, querying and EM training wuta are probably of instrumental value also for plan recogntion.

The set of plan operators considered in this paper is simplistic, but it suffices to illustrate the interaction between syntax, semantics, and computation within the wuta framework. A richer set is needed for practical usage, and this goes beyond syntactic sugar. With the present set of operators, the analyst modelling a plan can only express that two intermediate goals can be realised in any order, but not that they can be realised in parallell with interleaved observation sequences. Expressiveness thus increases if a shuffle oper-

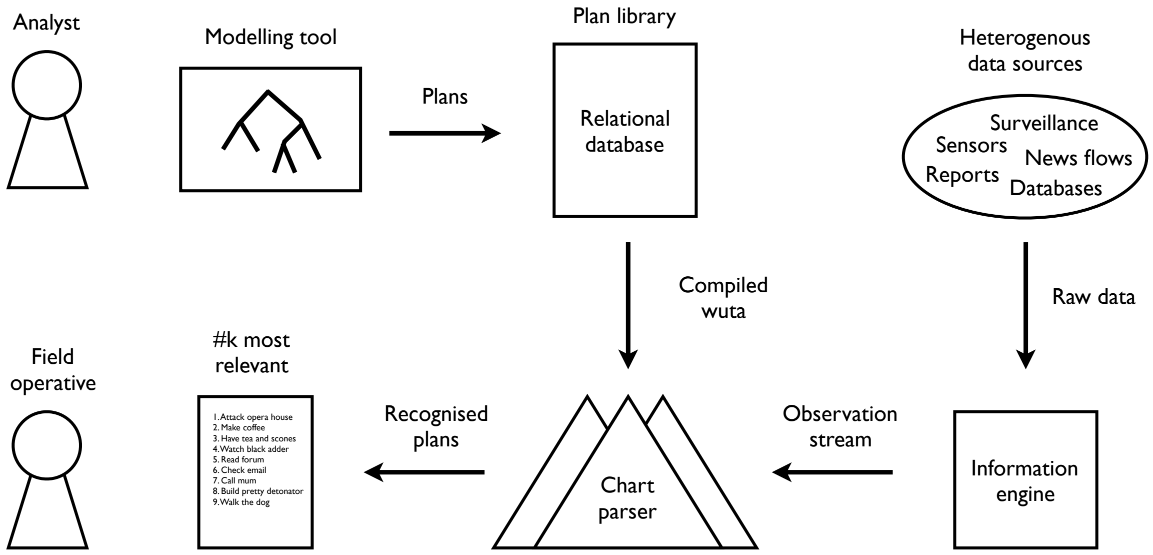

Figure 2: The user works with a modelling tool to create and edit plans. The plans are stored in a plan library and compiled upon demand into a wuta. The wuta can the be used to parse observations streams and produce a list of the k plans that are most likely to be active in a given moment.

ation is added the operator set, but this makes parsing computationally expensive and must be balanced by restrictions on the plan library. It holds generally that every addition to the operator set is best proceeded by an investigation of its effect on parsing complexity.

After the algorithmic toolkit and set of plan operators have been decided upon, implementation of an Impactorium component based on the outline in Figure 2 can commence. Tests will be conducted for military intelligence as well as for supply chain security and point security scenarios.

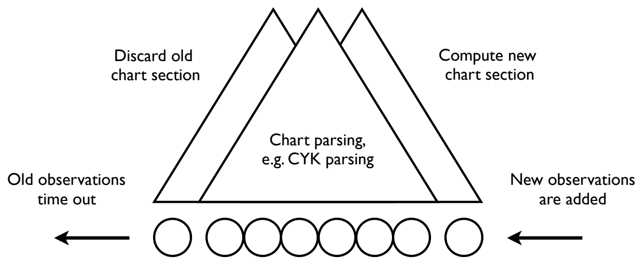
Discard old chart section
Compute new chart section
Chart parsing, e.g. CYK parsing
Old observations time out
New observations are added

Figure 3: Traditional chart parsing can be adapted to process a stream of observations, by adding chart sections when new observations are made, and discarding chart sections when observations become obsolete.

References

[1] P. A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular tree model checking. In *Proc. CAV'02*, volume 2404 of *LNCS*, 2002.

[2] D. W. Albrecht, I. Zukerman, A. Nicholson, and A. Bud. Towards a Bayesian model for keyhole plan recognition in large domains. In *Proc. 6th Int. Conf. on User Modeling*, pages 365–376. Springer-Verlag, 1997.

[3] D. Avrahami-Zilberbrand and G. A. Kaminka. Fast and complete symbolic plan recognition. In *Proc. IJCAI'05*, 2005.

[4] D. Avrahami-Zilberbrand and G. A. Kaminka. Utility-based plan recognition: An extended abstract. 2007.

[5] G. Barton. On the complexity of ID/LP parsing. *Comp. Linguistics*, 11:205–218, 1984.

[6] B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Proc. CAV'94*, volume 818 of *LNCS*, pages 55–67. Springer, 1994.

[7] W. S. Brainerd. The minimalization of tree automata. *Inform. and Control*, 13(5):484–491, 1968.

[8] H. H. Bui. A general model for online probabilistic plan recognition. In *Proc. IJCAI'03*, pages 1309–1315, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[9] H. H. Bui, S. Venkatesh, and G. West. Policy recognition in the abstract hidden markov model. *J. Artif. Int. Res.*, 17(1):451–499, 2002.

[10] S. Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48, 2001.

[11] J. Carme, J. Niehren, and M. Tommasi. Querying unranked trees with stepwise tree automata. In *RTA*, volume 3091 of *LNCS*, pages 105–118, 2004.

[12] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997. Release October, 1rst 2002.

[13] J. E. Doner. Decidability of the weak second-order theory of two successors. *Notices of the American Math. Society*, 12:365–468, 1965.

[14] J. E. Doner. Tree acceptors and some of their applications. *J. Comput. Syst. Sci.*, 4:406–451, 1970.

[15] J. d'Orso and T. Touili. Regular hedge model checking. In *IFIP TCS*, pages 213–230, 2006.

[16] F. Drewes. MAT learners for recognizable tree languages and tree series. *Acta Cybernetica*, 19:249–274, 2009.

[17] M. Droste, W. Kuich, and H. E. Vogler. *Handbook of Weighted Automata*. Monographs in Theoret. Comp. Sci. 2009.

[18] S. Eilenberg. *Automata, Languages, and Machines – Volume A*, volume 59 of *Pure and Applied Math.* Academic Press, 1974.

[19] J. Engelfriet, S. Maneth, and H. Seidl. Deciding equivalence of top-down xml transformations in polynomial time. *J. Comput. Syst. Sci.*, 75(5):271–286, 2009.

[20] R. Forsgren, L. Kaati, C. Mårtenson, P. Svenson, and E. Tjörnhammar. An overview of the Impactorium tools. 2008.

[21] C. W. Geib and S. A. Harp. Emperical analysus of a probalistic task tracking algorithm. In *MOO-04*, 2004.

[22] C. W. Geib and M. Steedman. On natural language processing and plan recognition. In *IJCAI*, pages 1612–1617, 2007.

[23] J. Golan. *Semirings and their applications*. Kluwer Academic, 1999.

[24] J. Högberg, A. Maletti, and H. Vogler. Bisimulation minimisation of weighted automata on unranked trees. *Fundam. Inf.*, 92(1-2):103–130, 2009.

[25] L. Huang and D. Chiang. Better k-best parsing. In *Proc. 9th Int. Workshop on Parsing Technology*, pages 53–64, Morristown, NJ, USA, 2005. Assoc. for Comp. Linguistics.

[26] E. M. C. Jr., O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[27] H. Kautz. *A Formal Theory of Plan Recognition*. PhD thesis, Dept. of Comp. Sci. University of Rochester, 1987.

[28] H. A. Kautz and J. F. Allen. Generalized plan recognition. In *AAAI*, pages 32–37, 1986.

[29] C. Koch. Efficient processing of expressive node-selecting queries on xml data in secondary storage: A tree automata-based approach. In *VLDB*, pages 249–260, 2003.

[30] P. Krauthausen and U. D. Hanebeck. Intention recognition for partial-order plans using dynamic Bayesian networks. In *Inform. Fusion*, 2005.

[31] M. Lohrey and S. Maneth. The complexity of tree automata and xpath on grammar-compressed trees. *Theoret. Comp. Sci.*, 363(2):196–210, 2006.

[32] M. Maragoudakis, A. Thanopoulos, and N. Fakotakis. MeteoBayes: Effective plan recognition in a weather dialogue system. *IEEE Intel. Sys.*, 22:67–77, 2007.

[33] W. Martens and J. Niehren. On the Minimization of XML-Schemas and Tree Automata for Unranked Trees. *J. Comput. Syst. Sci.*, 73:550–583, 2007.

[34] D. McMichael and G. Jarrad. Grammatical methods for situation and threat analysis. 2005.

[35] D. McMichael, G. Jarrad, S. Williams, and M. Kennett. Modelling, simulation and estimation of situation histories. pages 928–935, 2004.

[36] J. Mezei and J. B. Wright. Algebraic automata and context-free sets. *Inform. and Control*, 11(1):3–29, 1967.

[37] D. Pynadath. *Probabilistic Grammars for Plan Recognition*. PhD thesis, University of Michigan, 1999.

[38] D. Pynadath and M. Wellman. Accounting for context in plan recognition, with application to traffic monitoring. In *Proc. UAI'95*, pages 472–48. Morgan Kaufmann, 1995.

[39] M. Ramírez and H. Geffner. Plan recognition as planning. In *IJCAI*, pages 1778–1783, 2009.

[40] W. C. Rounds. *Trees, transducers, and transformations*. PhD thesis, Stanford University, 1968.

[41] W. C. Rounds. Mappings and grammars on trees. *Math. Sys. Theory*, 4(3):257–287, 1970.

[42] C. Schmidt, N. Sridharan, and J. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *A.I.*, 11(1,2), 1978.

[43] M. Schützenberger. On the definition of a family of automata. *Inform. and Control*, 4:245–270, 1961.

[44] T. Schwentick. Automata for xml - a survey. *J. Comput. Syst. Sci.*, 73(3):289–315, 2007.

[45] M. Sipser. *Introduction to the Theory of Computation*. International Thomson Publishing, 1996.

[46] J. W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *J. Comput. Syst. Sci.*, 1:317–322, 1967.

[47] J. W. Thatcher and J. B. Wright. Generalized finite automata with an application to a decision problem of second-order logic. *Math. Sys. Theory*, 2:57–82, 1968.

[48] A. Waern and O. Stenborg. A simplistic approach to keyhole plan recognition. SICS Technical Report T95:01, SICS, 1995.