Presented at the AAAI Workshop on Plan, Activity and Intent Recognition, February 2018.

Plan Recognition for Network Analysis: Preliminary Report

Robert P. Goldman Scott E. Friedman
Jeffrey M. Rye
SIFT, LLC
Minneapolis, MN USA
{rpgoldman,sfriedman,jrye}@sift.net

1 Abstract

In this paper, we describe the PRIMROSE system, which uses plan recognition to feed network analysis. PRIMROSE uses the probabilistic plan recognition algorithm YAPPR to process streams of reports. For these report streams, YAPPR generates plan trees as hypotheses. From the activities and role-fillers (entities that participate in these activities, such as agent, patient, destination, etc.) in the plan trees, YAPPR populates organizational graphs, intended to feed network analysis. The graph population is performed using a combination of rules for role extraction and analogical reasoning. We describe the methods used for plan recognition and role extraction. We also introduce the simulated narcotics trafficking domain, which we have implemented using an enhanced version of Blaylock's Monroe system. Contributions of this work include: (1) extensions to both the Monroe plan-based activity simulator, which has been modified to be agent-centric, and domain-configurable; (2) extensions to YAPPR to process (finite) first-order domains, and build plan trees; (3) the use of role-based reasoning to infer network structure; (4) employing analogical reasoning to infer coreference across plans.

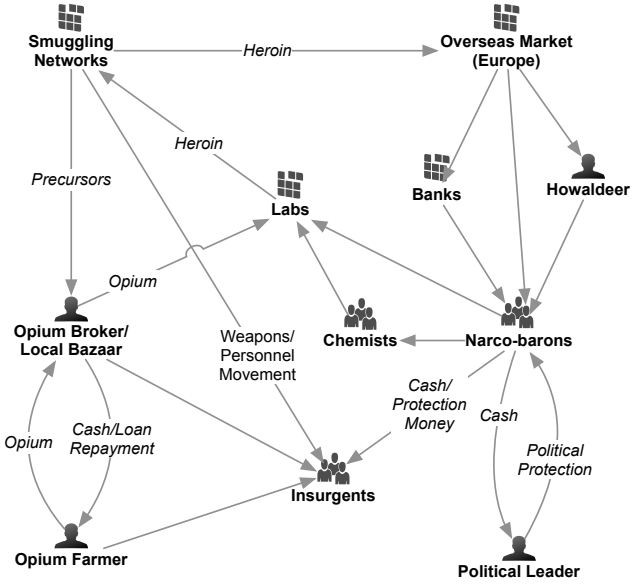

Figure 1: Example network analysis (taken from (DoD 2009)).

2 Introduction

Intelligence agencies often apply network analysis to their problems (Sparrow 1991; DoD 2009, *e.g.*), and must identify those networks by (reports of) observations. An example of such a network analysis is given in Figure 1. Network analyses like these are used as frameworks for organizing information (*e.g.*, identifying the participants in a narcotics trafficking network run by a particular narco-baron) and planning countermeasures (*e.g.*, hampering the manufacture of heroin by arresting heroin chemists). In this paper, we describe the PRIMROSE system whose purpose is to take reported actions, recognize underlying plans using probabilistic plan recognition, and based on the recognized plans, populate and maintain a network analysis like that in Figure 1. For example, reports of truck drivers making deliveries and pickups at known heroin labs will provide information about the makeup of the smuggling network(s) that support the narco-baron that operates the lab in question. PRIMROSE will help intelligence (and other) analysts process large amounts of information, and avoid errors in reasoning, by representing the *provenance* of information in the network analysis.

PRIMROSE's system architecture is shown in Figure 2. PRIMROSE uses probabilistic plan recognition – in particular, an enhanced version of the YAPPR algorithm (Geib and Goldman 2009; Geib, Maraist, and Goldman 2008; Maraist, Geib, and Goldman 2009) – to process streams of observations, and use the output of plan recognition to populate graphs for network analysis. Specifically, PRIMROSE operates in a simulated narcotics trafficking domain, implemented using an extension of the Monroe system (Blaylock and Allen 2005). PRIMROSE processes observations of narcotics trafficking activities, and innocent activities that act as confounds. From the resulting plan trees, PRIMROSE extracts key role-fillers and uses them to populate network analysis graphs, identifying the structure of narcotics trafficking organizations. This extraction is performed using a combination of rule-based inference and analogical reasoning.

Contributions of this work include: (1) extensions to both the Monroe plan-based activity simulator, which has been modified to be agent-centric, and domain-configurable; (2) extensions to YAPPR to process (finite) first-order domains, and build plan trees; (3) the use of role-based reasoning to infer network structure; (4) using analogical reasoning to infer coreference across plans.

3 PRIMROSE Architecture

PRIMROSE's system architecture is shown in Figure 2.

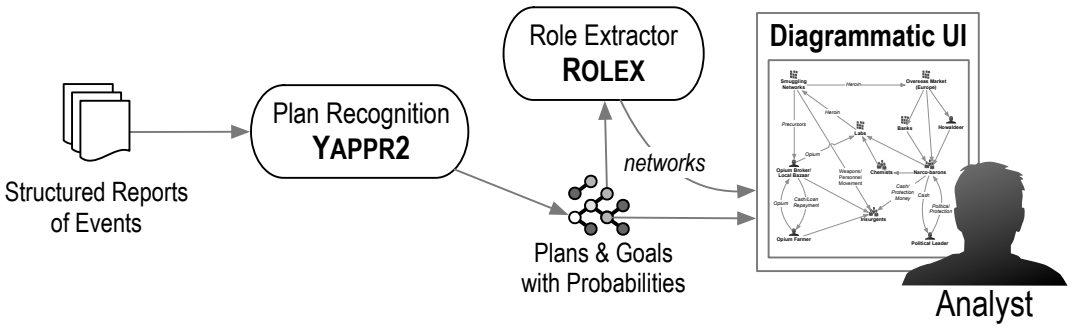

Figure 2: PRIMROSE system architecture.

PRIMROSE comprises the YAPPR2 plan recognition component, built on our previously-existing YAPPR system, and the Role Extraction (ROLEX) component. The ROLEX component, in turn is built on SIFT's SPIRE domain-independent reasoning system. We briefly review the capabilities of SPIRE here; we discuss YAPPR further below.

The central data storage, inference, and user interface of PRIMROSE is built on top of SIFT's SPIRE system. SPIRE is SIFT's domain-general graph fusion and domain-independent reasoning system. SPIRE natively supports graph-matching, to enable graph-based data fusion, domain-general comparative analysis (McLure, Friedman, and Forbus 2015), graph-based generalization (Friedman 2015), graph-based inference (McDonald et al. 2016), and similarity-based retrieval of events and entities. Given a semantic graph— such as a link diagram or output of a semantic parse— SPIRE can retrieve highly-isomorphic graphs, match graphs to fuse knowledge, and generalize abstract schemas from multiple examples.

SPIRE also includes an implementation of the *Assumption-based Truth Maintenance System (ATMS)* (deKleer 1986; Forbus and deKleer 1993), which we have used as the framework for implementing provenance reasoning (Section 7, below). The rule- and graph- matching features are used in ROLEX (Section 6), to support entity extraction from plans and population of network analyses.

SPIRE includes a web-based user interface (Figure 3) for navigating complex networks using diagram-based interaction.

4 The Simulated Domain

PRIMROSE takes its input from a simulation of narcotics trafficking in the Badakhshan province of Afghanistan. The simulation was built on an extension of Blaylock's Monroe system for stochastic plan simulation (Blaylock and Allen 2005). We extended Monroe to make it agent-centric and domain configurable. We drew upon UN reports on drug trafficking in Afghanistan (Demirbüken et al. 2011; Demirbüken, Mili, and Cussan 2012), and an example network analysis of a narcotics trafficking network (DoD 2009). To make our model more accurate, we also drew on background material about Afghanistan (UN Office for the Coordination of Humanitarian Affairs (UNOCHA) 2014; Megerdoomian 2009), and additional geographic information from the Falling Rain website,[1].

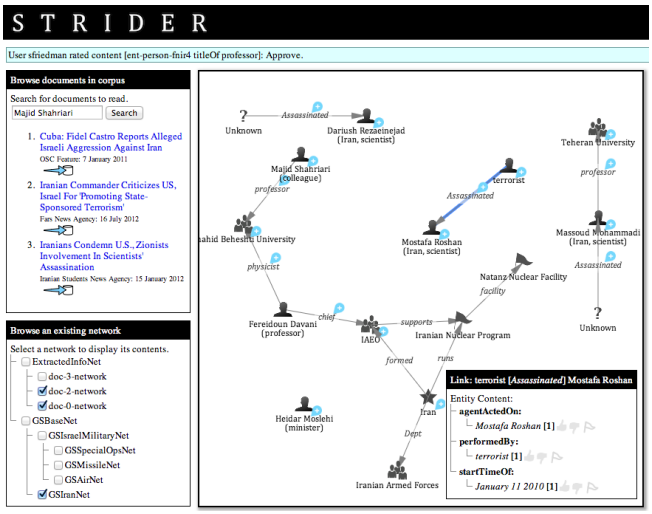

Figure 3: UI for SIFT's STRIDER system, built with SPIRE.

Blaylock's Monroe system uses a version of the SHOP2 hierarchical task network (HTN) planner (Nau et al. 2001; 2003; 2004) that supports random plan generation, as the core of a system that can generate large corpora of simulated plan-driven activity in a disaster relief domain (Blaylock and Allen 2005). Monroe randomly chooses goals from its distribution of top level goals, then uses SHOP2 to randomly generate a plan that meets that goal. For plan recognition research projects, Monroe has the advantage of providing large corpora labeled with ground truth.

For our PRIMROSE experiments, we made a number of extensions to the original version of Monroe to make our own enhanced Monroe, which we will call "Monroe2." First, we updated Monroe2 from using its own, built-in version of SHOP2, and replaced it with the mainstream version of SHOP2[2]. This had the dual advantages of bringing updates and bug fixes from SHOP2 into Monroe, and of bringing random search to SHOP2 (since we had to augment SHOP2 to replicate the randomized search that Blaylock had

[1] www.fallingrain.com

[2] www.sourceforge.net/project/shop/

implemented).

Secondly, Monroe2 is what we call "agent-centric." By this we mean that instead of simply randomly choosing a top-level goal as in the original Monroe, Monroe2 first chooses an agent, and then chooses an goal based on the agent type. Specifically, Monroe2 allows the modeler to specify a probability distribution over agent types, and within a type, over the agents of that type. Then, for each agent type, Monroe2 consults a separate probability distribution over goal types. See Figure 4 for a comparison between the two methods. This agent-centric nature is critical for our application, because different agents will have different types of goals, so it is natural factor this as randomly choosing an agent first, and then a goal that is appropriate to that agent. In the original Monroe there is conceptually only a single agent that is being observed.

The original Monroe specified a probability distribution over initial states as a function of the goal type. Monroe2 extends this to partition the initial state into a problem-independent and problem-dependent component, which are generated (stochastically) separately: the problem-independent state is shared across all of the plans for a corpus. This is critical for our application, because PRIMROSE will accumulate information over an entire corpus of plans, so that information must be consistent over the corpus. Note that this also means that Monroe2 has a notion of "domain setting" that is higher order than that of corpus, and from a single domain, can generate multiple corpora for different domain settings. Finally, Monroe2 separates out a notion of domain from the code itself: the original Monroe was domain-specific. Fortunately, the original code was very cleanly structured and this extension was not too difficult.

We hope to make Monroe2 and our narcotics trafficking domain model publicly available at the end of the PRIMROSE project. Doing so primarily requires making a set of user documentation to make it generally useful.

5 Plan Recognition

For PRIMROSE, we adopt the YAPPR framework for probabilistic plan recognition (Geib and Goldman 2009; Geib, Maraist, and Goldman 2008; Maraist, Geib, and Goldman 2009). *Probabilistic* plan recognition is critical in fields like intelligence analysis, because a key part of the task is for analysts to identify interesting plans, and filter out less interesting, but more likely plans (that truck driver isn't delivering opium to a heroin lab; they are just delivering cement to a compound for construction).

YAPPR is based on a probabilistic model of the execution of simple hierarchical plans (Erol, Hendler, and Nau 1994). In this hierarchical framework, tasks (goals) are decomposed into sub-tasks (sub-goals) in order to build a plan. In this framework, plan libraries are partially ordered AND/OR trees. AND-nodes represent *methods* for achieving a particular task: all of the children of an AND-node must be performed in order to perform the parent task. The children may be further constrained to be performed in a particular order (or one of a set of possible orders), by annotating them with pairwise ordering constraints.

Given probabilities for the top level goals, and probabilities for different method choices (at the OR nodes), it is not complicated to define the posterior probability of a given hypothesis. To think about this in the simple language of probability theory, an agent chooses some number of top level goals to have, and methods for achieving those goals. This makes some set of actions available to the agent to execute, and this set of actions is updated, as time goes by, and performing earlier actions enables the performance of later actions. For example, in our narcotics problem, bringing the opium to the lab enables the action of manufacturing heroin from the opium. A description of the model in very abstract terms may be found in a UAI paper by the PI and colleagues (Goldman, Geib, and Miller 1999). This model is equivalent to a Hidden Markov Model with complex internal state – the complexity coming from the task-subtask structure in the agent's plans.

YAPPR adopts performance-improving techniques from parsing to better handle plan recognition (Geib, Maraist, and Goldman 2008; Geib and Goldman 2009). For this project, we enhanced the existing YAPPR implementation to support actions with attributes represented as parameters or roles. Previously, YAPPR could only recognize plans with actions and complex tasks that were propositional. This was essential so that we could have primitive and complex actions with roles that would make the role extraction possible.

We extended the grammars used by YAPPR to permit complex (first order) terms as both terminals (actions) and internal nodes (complex tasks). Arguments to these terms are constrained by coreference constraints. For example, a rule for smuggling items out of the local province by truck is as follows:

```
(truck-smuggle-out ?trucker ?truck ?item ?source ?
     crossing ?dest) ⟶
   (deliver ?item ?trucker ?truck ?source ?crossing) >
   (deliver ?item ?trucker ?truck ?crossing ?dest)
```

In this rule the `?trucker` smuggles material from the `?source` to the `?dest` by first delivering the `?item` to a border-crossing, `?crossing`, and then from the `?crossing` to the final destination, `?dest`. The > indicates an ordering relationship between the two subtasks. An example of a plan tree generated by YAPPR2 is given in Figure 5.

This is plan tree represents a plan for manufacturing heroin at a lab `LAB144` by an unknown baron `?BARON12981`, including sub-tasks for acquiring the chemist `Morad`, the chemist producing the heroin, and the acquisition of opium. Here is a subset of the generated plan tree, where other sub-tasks (e.g., acquiring acetic anhydride) are omitted to conserve space: In the plan tree, primitive actions (i.e., observables) are preceeded by `!`, and unobserved— but still hypothesized— entities are preceded by `?`, as in the case of `?BARON12981`. In the primitive `!DRIVE-TRUCK`, `Matteen` is the truck driver (the agent), `BDKTRK4`[3] is the truck (the patient), `Pas-e_Khir` the source, and `LAB144` the destination. In the complex action `DELIVER`, the patient is `opium`, the agent is again

[3] "Badakhshan truck 4."

```
proc generate_problem ()
;; goal schema is a term with free variables
let schema = pchoose(top-goals)
    state = fixed-state
    begin
      state := state ∪ random-locations(state)
      ;; final goal has the free variables filled
      let goal = instantiate(schema, state)
      return problem = <goal, state>
    end
```

(a) Monroe problem generation.

```
proc generate_problem (config)
let* type = pchoose(agentTypes(config)) ; agent type
     agent = pchoose(agents(type,config)) ; agent instance
     schema = pchoose(goals(type,config))
     state = fixed-state(config)
     begin
       state := state ∪ stateInit(config, state)
       let goal = instantiate(schema, state)
          return problem = <goal, state>
     end
```

(b) Monroe2 problem generation.

Figure 4: Monroe and Monroe2 problem generation contrasted.

```
((MANUFACTURE-HEROIN ?BARON12981 LAB144)
 (((ACQUIRE-CHEMIST Morad LAB144)
   ((!TRAVEL Morad Mahall-i-Kamar LAB144)))
  ((COOK-HEROIN Morad LAB144)
   ((!PRODUCE-HEROIN Morad LAB144)))
  ((ACQUIRE-HEROIN-INGREDIENTS LAB144)
   (((ACQUIRE-HEROIN-INGREDIENTS-NEED-BOTH LAB144)
     (((ACQUIRE-OPIUM LAB144)
       (((DELIVER OPIUM Matteen BDKTRK4 Pas-e_Khir LAB144)
         (((GET-TRUCK-THEN-DELIVER OPIUM Matteen BDKTRK4 Bazar-
              e_Tashkan Pas-e_Khir LAB144)
           (((UNLOAD-TRUCK Matteen BDKTRK4 LAB144 OPIUM)
             ((!UNLOAD-TRUCK Matteen BDKTRK4 LAB144 OPIUM)))
            ((DRIVE-TRUCK Matteen BDKTRK4 Bazar-e_Tashkan Pas-
                 e_Khir)
             ((!DRIVE-TRUCK Matteen BDKTRK4 Bazar-e_Tashkan Pas-
                  e_Khir)))
            ((LOAD-TRUCK Matteen BDKTRK4 Pas-e_Khir OPIUM)
             ((!LOAD-TRUCK Matteen BDKTRK4 Pas-e_Khir OPIUM)))
            ((DRIVE-TRUCK Matteen BDKTRK4 Pas-e_Khir LAB144)
             ((!DRIVE-TRUCK Matteen BDKTRK4 Pas-e_Khir LAB144)))))))
```

Figure 5: Example plan tree: subset of plan tree for a plan to manufacture heroin at a heroin lab.

`Matteen`, the truck used is `BDKTRK4`, and `Pas-e_Khir` and `LAB144` the source and destination, respectively.

In order to enable YAPPR2 to process such plans, we augmented it with data structures for terms and a unifier. Using these capabilities required an extension of the search algorithm to check unification constraints.

The final extension made in YAPPR2 was the generation of plan trees. Previously, YAPPR, like many other such systems, simply returned root goals as the output of plan recognition. This was not sufficient for the needs of role extraction (see the following section). Fixing this, in combination with the addition of first order terms, was somewhat difficult, since the original YAPPR code took advantage of the limitation to goal recognition, to optimize by discarding many intermediate data structures that needed to be retained for plan tree generation.

6 Role Extraction (ROLEX)

When plan recognition is complete, PRIMROSE's ROLEX component takes likely plan trees and from them extracts key role-fillers using a combination of rule-based and analogical reasoning. The actions in the plans, both observed primitive actions, and inferred sub-tasks, are represented as first-order terms, with roles/arguments such as agent, patient, source, destination, location, etc. Values for these arguments are filled by unification, both by matching against primitive terms, and by unification constraints in the plan library. Since some entities in the plan may not be observed (see example below), after plan recognition there may be unbound variables in the plan trees.

ROLEX operates incrementally on one or more plan trees at a time, with the following operations:

1. Encode the plan tree into triples with sub-task relations.
2. Create skolem terms for all unobserved entities.
3. Use the triple-graph of all plan trees seen so far to perform rule-based inference of roles.
4. Record provenance of all inferences in an ATMS.
5. Detect mergable skolem terms via graph-matching.

We briefly describe each of these processes next.

ROLEX encodes each task in a plan tree into triples using the positions of the arguments to assign predicates. For example, the task `(ACQUIRE-CHEMIST Morad LAB144)` in the plan tree of Figure 5 is assigned a unique symbol (e.g., `event-29234`), and encoded as follows:

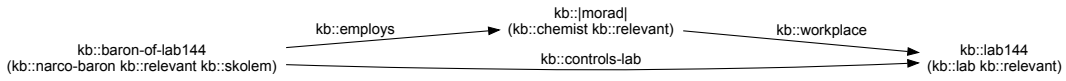

Figure 6: ROLEX infers roles from observed plans.

```
(isa event-29234 acquire-chemist)
(chemist event-29234 Morad)
(lab event-29234 LAB144)
```

Note that this involves assigning role names, such as `chemist` to relationships that, in YAPPR2, were represented only positionally. This also means that this representation can be stored as a graph in a triple store (*e.g.*, an RDF database or other graph DB).

For unobserved entities (unbound variables) such as `?BARON12981`, ROLEX constructs a functional skolem term based on other arguments within the action, to produce a concise symbol such as `BARON-OF-LAB144`. Since these terms are generated for functional relations, the same symbol will be retrieved when another unobserved baron is associated with `LAB144`, allowing ROLEX to accumulate information about hypothesized entities, absent direct observation.

ROLEX's rule-based inference combines Horn clauses and forward-chaining rules. For example, the following rule uses the two adjacent Horn clauses to infer that a baron employs a chemist when the chemist uses a lab in any event, and the baron uses the same lab in any event:

```
(RULE :lhs ((chemist-uses-lab ?lab ?chemist)
            (baron-uses-lab ?lab ?baron))
      :rhs ((employs ?baron ?chemist)))

(HORN :head (chemist-uses-lab ?lab ?chemist)
      :body ((lab ?event ?lab)
             (chemist ?event ?chemist)))

(HORN :head (baron-uses-lab ?lab ?baron)
      :body ((lab ?event ?lab)
             (baron ?event ?baron)))
```

These and other rules concisely describe role extraction patterns, allowing ROLEX to extract roles shown in Figure 6, which renders the inferred `employs` triple as a link. ROLEX encodes all of its logical inferences with rules and Horn clauses within an Assumption-based Truth Maintenance System (ATMS) (deKleer 1986; Forbus and De Kleer 1993). PRIMROSE subsequently uses the ATMS for provenance reasoning, so that all of ROLEX's inferences are defeasible if the supporting reports lose credibility, as described in Section 7. ROLEX's processing of these rules is sound and complete (because of the limitation to Horn clauses), which means that the extraction process is sound and complete *subject to the limitation of its rule set*. In future work we hope to use machine learning to acquire new ROLEX rules from experience and user feedback.

Over time, ROLEX extends its network by processing additional plan trees, as more observations are made and more plans are recognized. In our example, it processes additional action reports that indicate that `Morad` also produces heroin at `LAB143` which is operated by an unknown baron `BARON-OF-LAB143`, as shown in Figure 7. Further, ROLEX has inferred that both skolemized barons `BARON-OF-LAB144` and `BARON-OF-LAB143` are friendly to a the same smuggling network.

The skolem barons of `LAB143` and `LAB144` may refer to the same real-world individual, so ROLEX automatically suggests merging similar skolems via similarity-based reasoning. ROLEX computes relational semantic similarity via a greedy graph-matching algorithm (e.g., (Forbus et al. 2017)). It uses a *base* graph and a *target* graph (e.g., the subgraphs of `BARON-OF-LAB144` and `BARON-OF-LAB143`, respectively), and then greedily computes a *mapping* between the base and target. The mapping includes a similarity score s_m computed as a weighted node and edge count in the mapping, which it normalizes into the interval $[0, 1]$ by dividing it by the average node and edge count of the base s_b and target s_t: $s = \frac{2s_m}{s_b+s_t}$.

ROLEX includes a paramterizeable similarity threshold s^*, whereby it automatically suggests merging skolems if $s \geq s^*$ for any two same-typed skolems. This is the case for both barons in Figure 7. In our current work, s^* is set to 0.8. This was done for now by trial-and-error, but works robustly for this domain. This parameter could be adapted by machine learning, if needed.

Results from some preliminary experiments with ROLEX are shown in Figure 8. The plot represents the recall of narcotics network structure, measured against the ground truth from the Monroe2 configuration. There are two plots, one for raw recall, and one corrected for elements of the Monroe2 configuration that are impossible to observe. The results shown here represent essentially best-case results of ROLEX, since they are limited to the output of plan recognition with full action sequences, which removes most of the uncertainty from the plan recognition process. As such, this figure should be interpreted as showing *how well* ROLEX *could work if plan recognition was near optimal*. It should *not* be interpreted as a measure of overall PRIMROSE performance, whose evaluation awaits more extensive investigation.

7 Provenance reasoning

One key issue for analysts is the *provenance* (source) of the information that goes into their analyses. Using SPIRE, PRIMROSE will record the source of the observations that it receives as input. SPIRE, and thus PRIMROSE, model provenance using the Provenance Ontology for compatibility with other systems (Moreau et al. 2011). For example, use of the provenance ontology should assist in incorporating observations from other sources such as user judgments.

In PRIMROSE, provenance information is *propagated* through the inferences performed by YAPPR2 and ROLEX,

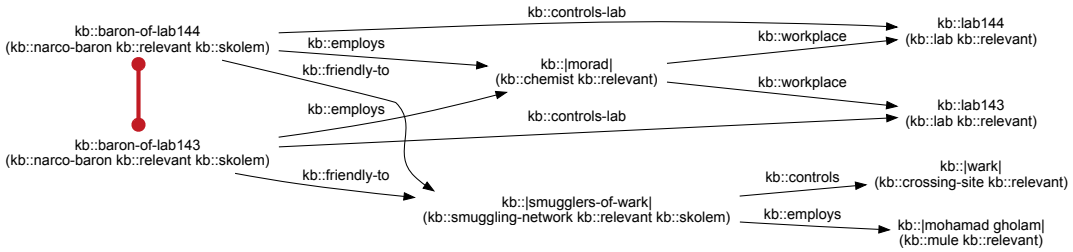

Figure 7: ROLEX uses graph-matching to infer sufficient similarity to merge two skolems.

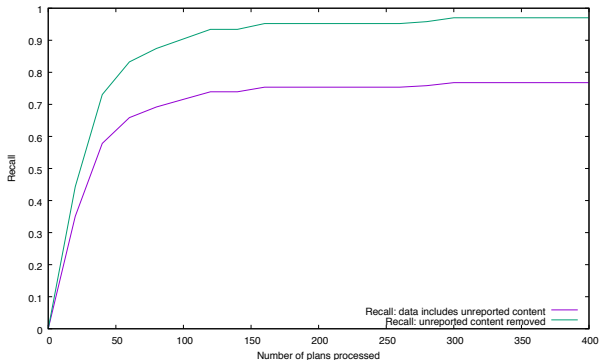

Figure 8: Recall performance of ROLEX as a function of number of plans processed.

using the *assumption-based truth maintenance system* (ATMS) (deKleer 1986; Forbus and deKleer 1993) in SPIRE. A truth maintenance system (TMS) (Doyle 1979) is a boolean constraint satisfaction system that records distinguished *assumptions*, and then propagates the assumptions to compute *labels* for every conclusion derived from those assumptions. An ATMS is a TMS that is distinguished by being able to maintain multiple different, possibly inconsistent *contexts* in a single boolean network (deKleer 1986; McDermott 1983). Each proposition in the ATMS graph will be so labeled, and the label gives the necessary and sufficient condition for that proposition to hold, in terms of the distinguished assumption nodes.

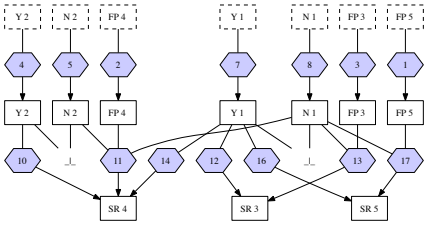

Figure 9: ATMS graph of inferences.

For example, Figure 9 shows an ATMS graph. The dotted rectangle nodes are *assumptions*, the solid rectangles *propositions*, and the hexagons are *justifications*: records of inferences. In this example, the justification nodes, 10, 11, and 12, that point to proposition SR4, indicate the implications $Y2 \rightarrow SR4$, $N2 \wedge N1 \rightarrow SR4$, and $Y1 \rightarrow SR4$, respectively. The assumptions in this network will be propagated to compute labels on all of the propositions. For example, the label of conclusion SR4 would be $\{\{Y1\}, \{Y2\}, \{N1, N2, FP4\}\}$. The ATMS efficiently computes these label sets, and can perform related tasks such as report all of the propositions that hold in a particular context, etc.

In PRIMROSE's ATMS, we add assumptions for each reported action, representing the assumption that the report is accurate. Reports also have informants, and we add assumptions to record the belief that the informant is credible. Reports of actions, then, are justified by the conjunction of the report accuracy assumption and the credible informant assumption. See Figure 11 for an example of these justification structures. Propositions about actions, such as `!travel-3440862613055040024-9999999` in the figure are justified by the reports and the credibility of the informant. Conclusions from plan recognition – for example in this case that there is an `acquire-chemist` complex action – is justified by the plan recognition process and, indirectly, the reports that went into it. Labels are propagated from these role-filler proposition through justifications that correspond to the rules that ROLEX uses to draw its further conclusions in the network analysis. See Figure 12 for an example.

Using provenance frameworks with the ATMS approaches allows browsable rationale for inferences, including the agent(s) responsible, the background knowledge responsible, the information sources responsible, etc. The combination of provenance and ATMS also allows PRIMROSE to readily *repudiate* an information source, and remove all of the conclusions supported by that information source — or, more precisely, all the conclusions supported by that information source *that do not have independent support from more reliable sources.* Because of its justification structure, PRIMROSE can adjust its conclusions either if an action report is incorrect (*e.g.*, an innocent mistake such as misreading a license plate), if it needs to retract all the conclusions based on a particular informant (*e.g.*, an informant is found to be abusing their position to settle grudges), or if an interpretation by YAPPR2 is discredited. For example, Figure 10 shows how information can be removed when an informant is discredited.

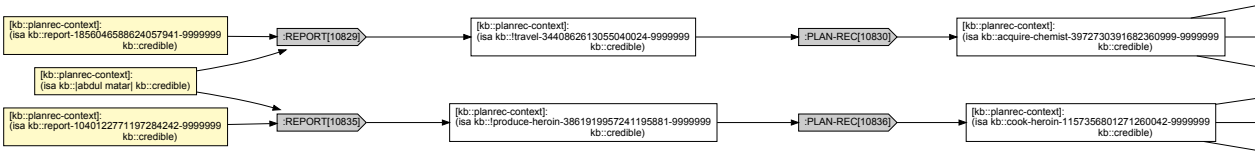

Figure 10: ATMS provenance structures for reports, showing dependency on assumptions about the report and informant, and justification of plan tree triples.

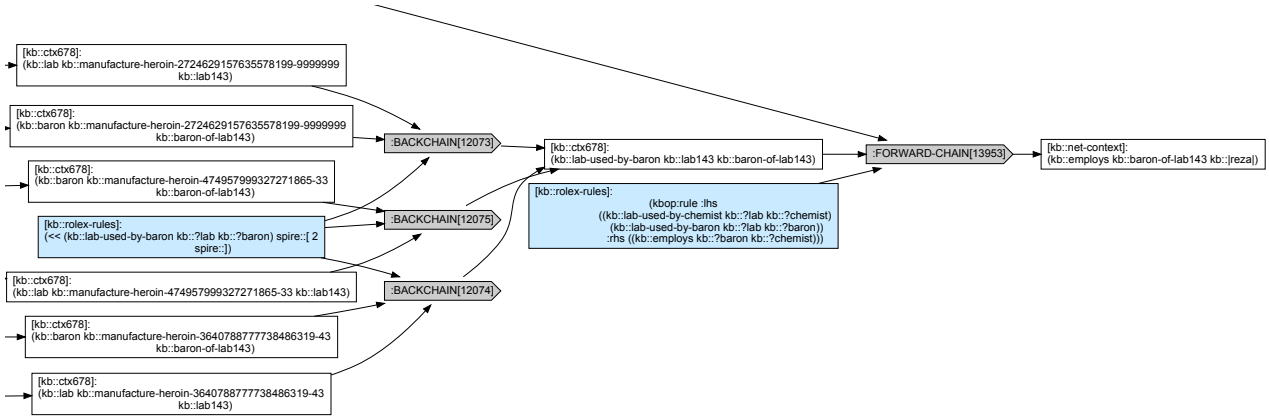

Figure 11: ATMS provenance structures for ROLEX's conclusions.

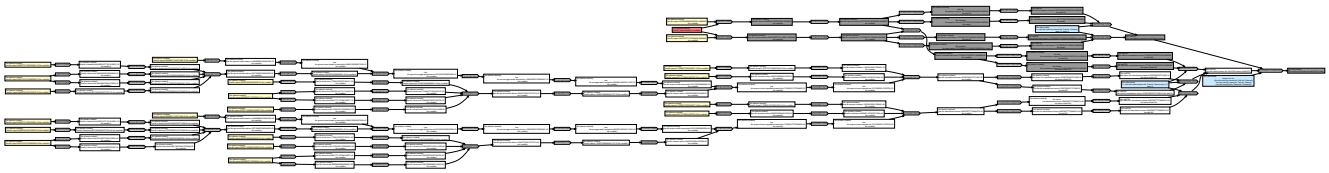

Figure 12: An ATMS network showing provenance for a set of PRIMROSE inferences. Grayed out nodes show how information can be removed when a source is discredited (red node).

8 Conclusions and Future Work

In this paper, we have described how our PRIMROSE system uses plan recognition in a pipeline for network analysis. Based on the roles that agents and objects play in observed plans, PRIMROSE populates a network analysis, using a combination of rule-based reasoning and analogy. PRIMROSE also tracks the provenance of information in the network analysis, allowing it to automatically remove inferences fed by discredit reports and reporters.

The work that we describe here is strictly a proof-of-concept. While we have a large quantity of corpora to analyze, we have not yet had time to run large scale experiments. For these experiments, we will need to make further improvements to YAPPR2. Efficiency improvements, like those developed by Geib for Elexir, and Kabanza for DOPLAR will be essential for PRIMROSE to scale up. By the time of the workshop, we hope to have empirical results to present, and to be able to release both Monroe2 and YAPPR2. In the longer term, we hope to use machine learning to automate the configuration of both YAPPR2 and ROLEX.

Acknowledgments This paper was supported by the Defense Threat Reduction Agency under contract number HDTRA1-17-P-0011 Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Blaylock, N., and Allen, J. F. 2005. Generating artificial corpora for plan recognition. In Ardissono, L.; Brna, P.; and Mitrovic, A., eds., *User Modeling*, volume 3538 of *Lecture Notes in Computer Science*, 179–188. Springer.

deKleer, J. 1986. An assumption-based TMS. *Artificial Intelligence* 28:127–162.

Demirbüken, H.; Mili, H.; Spassova, Y.; Azizi, H.; and Pashtoon, S. J. 2011. The global Afghan opium trade. Technical report, United Nations Office on Drugs and Crime.

Demirbüken, H.; Mili, H.; and Cussan, R. L. 2012. Opiate flows through northern Afghanistan and Central Asia. Technical report, United Nations Office on Drugs and Crime.

2009. *Joint Intelligence Preparation of the Operational Environment*.

Doyle, J. 1979. A truth maintenance system. *Artificial Intelligence* 12(3):231–272.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. UMCP: A sound and complete procedure for hierarchical task network planning. In Hammond, K. J., ed., *Artificial Intelligence Planning Systems:Proceedings of the Second International Conference*, 249–254. Waltham, MA: Morgan Kaufmann.

Forbus, K. D., and De Kleer, J. 1993. *Building problem solvers*, volume 1. MIT press.

Forbus, K. D., and deKleer, J. 1993. *Building Problem Solvers*. Cambridge, Massachusetts: MIT Press.

Forbus, K. D.; Ferguson, R. W.; Lovett, A.; and Gentner, D. 2017. Extending sme to handle large-scale cognitive modeling. *Cognitive Science* 41(5):1152–1201.

Friedman, S. E. 2015. Exploiting graph structure to summarize and compress relational knowledge. In *Proceedings of the 28th International Workshop on Qualitative Reasoning*.

Geib, C. W., and Goldman, R. P. 2009. A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence* 173(11):1101–1132.

Geib, C. W.; Maraist, J.; and Goldman, R. P. 2008. A new probabilistic plan recognition algorithm based on string rewriting. In Rintanen, J.; Nebel, B.; Beck, C.; and Hansen, E., eds., *Proceedings of the International Conference on Automated Planning and Scheduling*, 91–98.

Goldman, R. P.; Geib, C. W.; and Miller, C. A. 1999. A new model of plan recognition. In Laskey, K. B., and Prade, H., eds., *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, 245–254. S.F., Cal.: Morgan Kaufmann Publishers.

Maraist, J.; Geib, C. W.; and Goldman, R. P. 2009. On plan recognition and parsing. In Geib, C. W.; Pynadath, D. V.; Bui, H.; and Sukthankar, G., eds., *Workshop on Plan, Activity and Intent Recognition (PAIR)*.

McDermott, D. V. 1983. Contexts and data dependencies: A synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5(3):237–246.

McDonald, D.; Friedman, S.; Paullada, A.; Bobrow, R.; and Burstein, M. 2016. Extending biology models with deep NLP over scientific articles. In *Proceedings of the AAAI-16 Workshop on Knowledge Extraction from Text*.

McLure, M. D.; Friedman, S. E.; and Forbus, K. D. 2015. Extending analogical generalization with near-misses. In *AAAI*, 565–571.

Megerdoomian, K. 2009. The structure of Afghan names. Technical Report MP090315, MITRE.

Moreau, L.; Clifford, B.; Freire, J.; Futrelle, J.; Gil, Y.; Groth, P.; Kwasnikowska, N.; Miles, S.; Missier, P.; Myers, J.; et al. 2011. The open provenance model core specification (v1. 1). *Future generation computer systems* 27(6):743–756.

Nau, D. S.; Cao, Y.; Lotem, A.; and Muñoz-Avila, H. 2001. The SHOP planning system. *AI Magazine*.

Nau, D.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *JAIR* 20:379–404.

Nau, D.; Au, T.-C.; Ilgami, O.; Kuter, U.; Muñoz-Avila, H.; Murdock, J. W.; Wu, D.; and Yaman, F. 2004. Applications of SHOP and SHOP2. Technical Report CS-TR-4604, UMIACS-TR-2004-46, Computer Science Department, University of Maryland, College Park, MD.

Sparrow, M. K. 1991. The application of network analysis to criminal intelligence: An assessment of the prospects. *Social Networks* 13:251–274.

UN Office for the Coordination of Humanitarian Affairs (UNOCHA). 2014. Badakhshan Province: District atlas. Technical report, UNOCHA.