Medical Procedure Tracking using Abductive Planning - A PARADIGM Shift

Michael Wessel, Michael Cogswell, Jason Tyan, and Bob Price

SRI International

Abstract

This paper presents PARACHUTE, a Prolog-based framework for abductive trace completion and projection for real-time medical procedure tracking and guidance, developed as part of the AI-powered AMIRA system in the ARPA-H PARADIGM program. PARADIGM aims to expand access to high-quality healthcare in rural America by upskilling healthcare workers. Our key contribution is a neuro-symbolic framework that combines multimodal perception with abductive reasoning to robustly infer and guide procedures under noisy and incomplete observations. We describe the challenges of procedure modeling and tracking, and present our solutions for representing and reasoning about complex clinical workflows using abductive logic programming. This work has not been previously published.

Introduction & Overview

The ARPA-H PARADIGM program aims to deliver high-quality mobile healthcare to rural communities and underserved areas across the United States. A key technical focus of the program is the development of an AI-powered Intelligent Task Guidance (ITG) system that enables users to perform complex clinical procedures with expert-level accuracy: AMIRA is our ITG system. AMIRA integrates multimodal perception (vision, audio, and contextual sensing) with neuro-symbolic reasoning and large language models (LLMs) to provide real-time, step-by-step guidance, proactive decision support, and robust error detection. This work advances human–AI teaming in healthcare by upskilling users, enhancing safety and consistency, and expanding access to high-quality care across diverse clinical settings. The ITG system will be deployed on a *medical van*, offering medical procedures on-site and on-demand to walk-in patients.

In this paper, we present the symbolic procedure reasoning component of AMIRA: PARACHUTE (*Prolog-based Activity Recognition via Abductive Controlled Hallucination Using Temporal Engine)*. Unlike our previous Hierarchical Task Network (HTN)-based work that pre-dates PARADIGM (Price et al. 2024), PARACHUTE uses abductive logic programming for more flexibility and expressivity. By explicitly and transparently defining the set of correct procedures, the formal model implements core guidance logic that is safe. Unlike purely data-driven approaches, our method explicitly models procedural constraints while remaining robust to noisy and incomplete observations.

Outline: We illustrate the procedure modeling and tracking problem with a blood draw procedure, then frame the problem in an abductive logic programming setting.

Contributions: (1) a neuro-symbolic procedure tracking framework integrating perception and abductive reasoning. (2) a flexible procedural model supporting branching, loops, and temporal constraints; (3) a real-time system for guidance and error detection in clinical workflows.

Problem: Procedure Modeling & Tracking

In the following, we will use the **Blood Draw Procedure** as our running example. It is expected that the PARADIGM van will support venipuncture. A typical high-level blood draw procedure has the following "steps" in Figure 1.

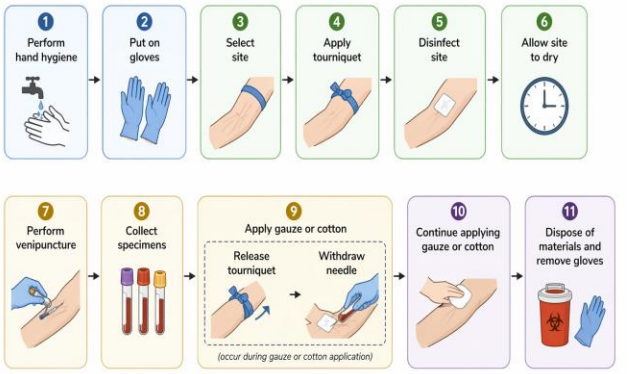

Figure 1: Simplified Blood Draw Procedure

Medical procedures are highly standardized, and the van clinicians are expected to adhere strictly to the gold procedure. AMIRA will monitor and guide the clinicians during blood draw execution, utilizing vision and speech as primary interaction modalities.

With its multiple stationary and moving video cameras, AMIRA is continually scanning for recognized actions (e.g., *apply_tourniquet, disinfect_site, …*) as well as for detected objects and spatial relations (e.g., o*n*(*gloves*1,*arm*1)).

Using this information, AMIRA should automatically inform the user that *disinfect_site* is next after it sees the tourniquet applied. It should also warn users if it sees them performing disinfection without wearing gloves.

Purpose & methodology of procedure tracking

Procedure tracking enables the following key capabilities:
a) it guides users through procedure execution by telling them where they are and suggesting what to do next.
b) it identifies deviations and errors in the observed procedure performance; for example, by informing the user of missing, out of order, or incorrectly executed steps.

Our procedure model supports both objectives by computing an internal representation that can best be understood as the *best-scoring hypothesis* of a valid procedure execution / **trace** (according to the model) given the observed evidence.

Because procedure observations are noisy and incomplete, tracking cannot rely on deduction alone: PARACHUTE uses abduction to hypothesize model-sanctioned actions and preconditions that must have occurred but were not observed. It computes a **projected trace completion**: a best-scoring, procedure-consistent execution hypothesis generated from the observed trace so far and the procedure model. The completed prefix explains the observed past, abducing missing evidence and flagging observations that cannot be reconciled with the model, while the projected suffix predicts likely future continuations through the procedure's next-step structure. Thus, the same representation supports retrospective error detection from the prefix and prospective next-step guidance from the suffix.

Challenges of medical procedure modeling

We require the following expressive means:

Branching: a splinting procedure may use plaster or fiberglass, and it is desirable to cover both variations in one common model (instead of having multiple models with duplicated content).

Iterations / loops / variables: some blood panel analyses require any number from 1 to 10 tubes of blood (vials) to be drawn from the patient. Having one model for each variation is again not desirable. Consequently, the procedure model must accommodate loops. Each step iteration can create new objects (e.g., filled blood vials) and unique names are needed to refer to these, and first-order variables are also desirable to refer to domain objects in general.

Temporal aspects: both *quantitative* and *qualitative* temporal relationships are required – for example, the *allow_site_to_dry* step after the disinfection will require 30 to 60 seconds (the alcohol needs to evaporate before we can continue). Moreover, *qualitative* temporal relations such as the *Allen temporal relations* (and variations, i.e., disjunctions thereof) are very useful for modeling. Consider that the tourniquet shall be removed *during* the application of the cotton swab to suppress bleeding of the puncture site.

Originally, we started with an HTN-based procedure representation (Tate 1977; Sacerdoti 1974; Wilkins 1984; Erol, Hendler, and Nau 1994), see Price et al. 2024, but the rigid commitment to a fixed task hierarchy in addition to the above listed expressivity challenges motivated the switch to a more flexible and expressive PDDL/STRIPS-inspired first-order representation (Fikes and Nilsson 1971; Fox and Long 2003), resulting in the Prolog-implemented **PARACHUTE Procedure Description Language (PPDL).**

PPDL is not a replacement for PDDL/PDDL2.1: it borrows lifted action schemas with PRE/ADD/DEL-style conditions and effects, but adds explicit *next*-step links, recovery/optional-step annotations, Prolog variables, and CLP(Q)/Allen temporal constraints used directly by PARACHUTE for abductive trace completion and next-step projection. Rich temporal planners can also express sophisticated temporal constraints over plans (Micheli and Scala 2019); our focus differs in that the procedure model is given and the main task is to complete a noisy, partially observed execution trace and project likely continuations online.

In medical procedures, the sequences of steps are largely pre-determined. This motivates explicitly modeling the *expected next steps,* instead of determining step applicability at runtime based on satisfied preconditions.

Challenges of medical procedure tracking

Procedure tracking is challenging because the *sensor data is noisy – false negatives* may result in "dropouts" in the observed procedure trace (was the step not performed or not detected?), and *false positives* introduce noise (was an incorrect or additional step executed?). A *robust* symbolic tracking procedure will have to address both issues.

Perception representation and system integration

Our PARACHUTE tracker assumes a fixed perception vocabulary, meaning all observables are known beforehand. This allows us to represent perceptions as *fluents,* or first-order predicates whose truth value can change over time. We use the term fluent for both the pre/postcondition atoms as well as for the temporally indexed atoms.

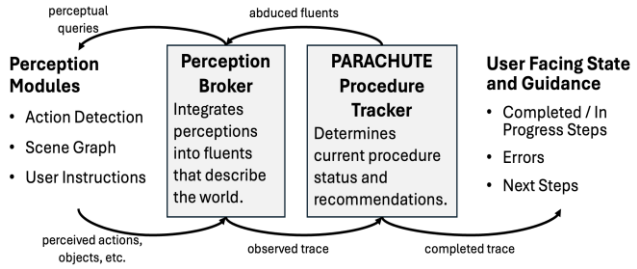

Figure 2: PARACHUTE in AMIRA.
The procedure tracker integrates through bottom-up and top-down interactions with perception.

As shown in Figure 2, the system guarantees this assumption through the perception broker. This module is responsible for translating raw perceptual messages to and from their fluent representation.

Procedure Definitions in PPDL

PPDL represents a procedure as a workflow-like next-step graph of first-order step schemas. Each step has an action predicate, STRIPS-like PRE, ADD, and DEL lists of fluents, a required/optional flag, possible successor steps, and optional temporal constraints. A procedure also designates a start/root step. Unlike HTNs, PPDL has no task-network hierarchy; however, steps may be tagged with virtual group memberships, e.g., *collect_tubes* may be grouped under a *prepare_supplies* super-step. Tracking itself is performed over leaf steps. For example:

```
step(id(disinfect_site),
     required(true),
     action(disinfect_site),
     pre([on(gloves1,arm1)]),
     add([site_disinfected]),
     del([]),
     next([allow_site_to_dry]),
     temporal([])).
```

For compactness, PPDL step definitions omit temporal arguments. These are introduced when step schemas are matched against observed actions or projected as future steps.

PARACHUTE performs **PPDL-based abductive trace completion and projection**. Following Gregory's idea that "vision is controlled hallucination" (Gregory 1980), it treats noisy observations as partial evidence for a larger procedure-consistent execution structure. Since tracking cannot be performed by deduction from observations alone (Shanahan 1989; Hobbs et al. 1993), PARACHUTE hypothesizes missing evidence, especially unobserved actions and preconditions, to construct projected trace completions that explain the observed prefix and project likely continuations (Poole, Goebel, and Aleliunas 1987; Denecker and Kakas 2002). This inference is controlled by the PPDL model: only model-sanctioned actions and fluents may be abduced, and candidate completions must satisfy the model's deductive and temporal constraints, extending earlier work on controlled abductive interpretation of visual and media observations (Möller, Neumann, and Wessel 1999; Espinosa Peraldi et al. 2007; Castano et al. 2009).

Input to PARACHUTE. The input is a partial, noisy observation trace O from perception, represented as a finite set of interval-timestamped atoms whose non-temporal object arguments are ground and whose temporal endpoints are either rational constants or CLP(Q) variables: action atoms $a(\bar{x}, t_s, t_e)$ and fluent atoms $p(\bar{x}, t_s, t_e)$. Here $\bar{x}$ denotes a possibly empty tuple of non-temporal object arguments (variables in step schemas and constants after grounding), while t_s, t_e denote temporal start and end terms handled by CLP(Q). Action atoms denote bounded events. Fluent atoms denote intervals over which conditions hold: a fluent with bound start and end times holds over that interval, while a fluent with a variable end time is open and holds until that variable is later constrained, for example by a deleting action (via DEL). For example,

{... *disinfect_site*(100,120), *on*(*gloves*1,*arm*1,80, T_g) ... }

states that the *disinfect_site* action occurred from 100 to 120 seconds and that the fluent *on*(*gloves*1,*arm*1) held from 80 seconds until the still-open time T_g. The precondition is satisfied if the fluent interval includes the action start time, here enforced by CLP(Q) constraints such as $80 \leq 100 \leq T_g$. If the trace instead contains

{... *disinfect_site*(100,120), *allow_site_to_dry*(120,180) ...}

but lacks evidence for the required precondition, PARACHUTE may abduce a missing fluent interval, for example

$$ab(on(gloves1, arm1, T_1, T_2))$$

together with temporal constraints $T_1 \leq 100 \leq T_2$, indicating that the gloves must have been on the arm when *disinfect_site* began. Similarly, ADD effects introduce derived fluent intervals. For example, *disinfect_site*(100,120) derives

site_disinfected(120,*Td*)

where the fluent starts holding at the action end time and remains open until T_d is constrained by a later deleting action, the trace horizon, or another temporal bound.

Output of PARACHUTE. The output is the top-k **projected trace completions**, each paired with its scoring vector. A projected trace completion is a PPDL-consistent step sequence plus observed, derived, and abduced action and fluent atoms, together with records of any observations that had to be disregarded. Its prefix explains the observed execution and supports error detection and perceptual verification; its suffix follows the procedure's next-step structure to project future continuations and support next-step recommendations. PARACHUTE computes these completions by Prolog backtracking: it aligns observations by unification, abduces missing actions or precondition fluents, may disregard observations that cannot be made consistent, checks CLP(Q)/Allen temporal constraints over action and fluent intervals, and returns the top-k completions under the scoring vector. This is related to abductive planning and plan-recognition-as-planning (Eshghi 1988; Ramírez and Geffner 2009), but here the search is procedure-guided rather than open-ended goal-directed planning.

Observed actions and fluents provide concrete interval endpoints when available. For projected future steps and unobserved fluents, endpoints may remain as variables constrained by the global temporal network. PARACHUTE requires subsequent step starts to be ordered, $t_{s1} < t_{s2} < \cdots < t_{sn}$, and each action to have positive duration, $t_{si} < t_{ei}$.

```
root_step(perform_hand_hygiene).

step(id(perform_hand_hygiene),
     action(perform_hand_hygiene),
     add([clean(hands)]),
     next([put_on_gloves])).

step(id(put_on_gloves),
     action(put_on_gloves),
     pre([clean(hands),
          not(on(gloves1,arm1))]),
     add([on(gloves1,arm1)]),
     next([select_site])).

step(id(select_site),
     action(select_site),
     pre([on(gloves1,arm1)]),
     add([site_selected]),
     next([apply_tourniquet])).

step(id(apply_tourniquet),
     action(apply_tourniquet),
     pre([site_selected]),
     add([applied(tourniquet1,arm1)]),
     next([disinfect_site])).

step(id(disinfect_site),
     action(disinfect_site),
     pre([site_selected,
          on(gloves1,arm1)]),
     add([site_disinfected]),
     next([allow_site_to_dry])).

step(id(allow_site_to_dry),
     action(allow_site_to_dry),
     pre([site_disinfected]),
     add([site_ready]),
     next([perform_venipuncture])).

step(id(perform_venipuncture),
     action(perform_venipuncture),
     pre([site_ready,
          on(gloves1,arm1)]),
     add([needle_in_vein]),
     next([collect_specimens])).

step(id(collect_specimens),
     action(collect_specimens),
     pre([needle_in_vein]),
     add([specimens_collected]),
     next([apply_gauze_or_cotton])).

step(id(apply_gauze_or_cotton),
     action(apply_gauze_or_cotton),
     pre([needle_in_vein]),
     add([gauze_applied]),
     next([release_tourniquet]),
     temporal([
        contains(release_tourniquet),
        contains(withdraw_needle)])).

step(id(release_tourniquet),
     action(release_tourniquet),
     pre([specimens_collected,
          tourniquet_applied,
          gauze_applied]),
     add([tourniquet_released]),
     del([tourniquet_applied]),
     next([withdraw_needle]),
     temporal([
        during(apply_gauze_or_cotton)])).

step(id(withdraw_needle),
     action(withdraw_needle),
     pre([needle_in_vein,
          tourniquet_released,
          gauze_applied]),
     add([needle_removed]),
     del([needle_in_vein]),
     next([terminal]),
     temporal([
        after_or_meets(release_tourniquet),
        during(apply_gauze_or_cotton)])).
```

Figure 3: Simplified PPDL blood draw model excerpt. Each block defines one PPDL step with action predicate, PRE/ADD/DEL, successor steps, and Allen constraints.

Additional Allen relations (Allen 1983) can be specified in the temporal field of step definitions.

Abduced past atoms are sent to the **perception broker** as expectations to verify, while the projected suffix forms a candidate procedure-guided continuation plan used for next-step recommendations.

Example Blood Draw Procedure

A more complete, but still simplified PPDL blood draw model is shown in Figure 3. The procedure utilizes Allen (1983) relations to express that both the tourniquet release as well as the needle withdrawal should happen during the application of the gauze (to prevent bleeding). Figure 4 shows a PARACHUTE invocation on an observation trace, and a subsequence of the completed trace and score.

The completed trace is a sequence of tuples with step id, optionality flag, times t_s and t_e, the action atom itself, precondition (PRE) and added (ADD) and deleted (DEL) fluents. The completed trace also includes *temporal information* from the observation trace - it was able to integrate the *apply_tourniquet*(100,120) and *site_selected*(80,*I*) observations into the completed trace. In particular, we find that *site_selected*(100,*I*) holds during *apply_tourniquet*(100,120), and that *ab(select_site*(*H*,80)) was "partially" abduced, as the post-condition fluent *site_selected*(80,I) provided the endpoint time for the action, yet the action itself was not observed in the call to `parachute`.

Scoring is a domain-configurable lexicographic penalty vector, not a probabilistic confidence.

```
QUERY
?- parachute([apply_tourniquet(100,120)],
             [site_selected(80,_)], Trace, WS, Score).

TOP-RANKED PROJECTED TRACE COMPLETION (EXCERPT)
put_on_gloves(D,E)      [ab(put_on_gloves(D,E))]
  pre: clean(hands,D,C); ab(not(on(gloves1,arm1,D,F)))
  add: on(gloves1,arm1,E,G)
select_site(H,80)       [ab(select_site(H,80))]
  pre: on(gloves1,arm1,H,G); add: site_selected(80,I)
apply_tourniquet(100,120)   [observed action]
  pre: site_selected(100,I); add: applied(tourniquet1,arm1,120,J)
                    -- projected continuation --
disinfect_site(K,L)     [projected; ab(disinfect_site(K,L))]
  pre: site_selected(K,I); on(gloves1,arm1,K,G)
  add: site_disinfected(L,M)

SCORE VECTOR
score{missing:1, missing_until_now:1, missing_until_now_vars:2,
      disregarded:0, abducibles:12, vars:21, trace:11,
      complexity:50, allen:0, nogoods:0}
```

Figure 4: **PARACHUTE query, projected trace completion, and score.** Given the observed action and fluent, it reconstructs earlier missing steps, aligns the observed tourniquet action, and projects a likely continuation.

By default, completions are preferred when they disregard fewer observations, require fewer abduced past actions/preconditions, leave fewer unresolved variables, and have lower trace/model-complexity terms; Figure 4 shows the currently exposed components. Thus top-k means the first k completions under this ordering. Here, the `disregarded:0` score indicates that the model was able to integrate all observations into the completed trace, as intended. Evidence should not be ignored if it can be integrated into a hypothesis.

The constructed world state is not shown here, but it is simply the aggregate of the (temporally indexed) fluents. In addition, the temporal constraint network and its corresponding CLP($\mathbb{Q}$) constraints between the temporal variables can be inspected (e.g., $E < G$ in the example).

Moreover, observations that cannot be made to fit the model can be disregarded. For example, consider:
{ *release_tourniquet*(100,120), *apply_tourniquet*(200, 300) }
There are three choices: either the release observation, the apply-tourniquet observation, or both can be disregarded. Lexicographic scoring can be redefined as needed for the domain.

Summary & Outlook

We presented PARACHUTE, a Prolog/PPDL-based online abductive procedure tracker for AMIRA, SRI's PARADIGM contribution. PARACHUTE computes projected trace completions that explain noisy observed prefixes and project next-step-guided future continuations, using bounded Prolog search with CLP(Q)/Allen temporal constraints. Preliminary use in the blood-draw model supports real-time error detection and next-step guidance; scalability over larger procedure libraries and uncertainty handling for noisy perception remain future work.

References

Allen, J. F. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26(11): 832–843.

Castano, S.; Espinosa Peraldi, I. S.; Ferrara, A.; Karkaletsis, V.; Kaya, A.; Möller, R.; Montanelli, S.; Petasis, G.; and Wessel, M. 2009. Multimedia Interpretation for Dynamic Ontology Evolution. *Journal of Logic and Computation* 19(5): 859–897.

Denecker, M.; and Kakas, A. C. 2002. Abduction in Logic Programming. In *Computational Logic: Logic Programming and Beyond. Essays in Honour of Robert A. Kowalski, Part I*, LNAI 2407, edited by A. C. Kakas and F. Sadri, 402–436. Berlin: Springer.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN Planning: Complexity and Expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 1123–1128. Menlo Park, CA: AAAI Press.

Eshghi, K. 1988. Abductive Planning with Event Calculus. In *Proceedings of the Fifth International Conference on Logic Programming (ICLP-88)*, edited by R. A. Kowalski and K. A. Bowen, 562–579. Cambridge, MA: MIT Press.

Espinosa Peraldi, S.; Kaya, A.; Melzer, S.; Möller, R.; and Wessel, M. 2007. Multimedia Interpretation as Abduction. In *Proceedings of the 2007 International Workshop on Description Logics (DL-2007)*, CEUR Workshop Proceedings Vol. 250.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2(3–4): 189–208.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20: 61-124.

Gregory, R. L. 1980. Perceptions as Hypotheses. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences* 290(1038): 181–197.

Hobbs, J. R.; Stickel, M. E.; Appelt, D. E.; and Martin, P. 1993. Interpretation as Abduction. *Artificial Intelligence* 63(1–2): 69–142.

Micheli, A.; and Scala, E. 2019. Temporal Planning with Temporal Metric Trajectory Constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 7675-7682.

Möller, R.; Neumann, B.; and Wessel, M. 1999. Towards Computer Vision with Description Logics: Some Recent Progress. In *Proceedings of the IEEE Workshop on Integration of Speech and Image Understanding*, 101–115.

Poole, D.; Goebel, R.; and Aleliunas, R. 1987. Theorist: A Logical Reasoning System for Defaults and Diagnosis. In *The Knowledge Frontier: Essays in the Representation of Knowledge*, edited by N. Cercone and G. McCalla, 331–352. Berlin: Springer-Verlag.

Price, B.; Chiou, Y.-M.; Crouch, D.; Youngblood, M.; Chen, J.; and Ortiz, C. 2024. Neuro-Symbolic Ensembles for Assistance at the Edge. In *Imaging and Multimedia Analytics at the Edge 2024*, IS&T Electronic Imaging Symposium. Society for Imaging Science and Technology.

Ramírez, M.; and Geffner, H. 2009. Plan Recognition as Planning. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-09)*, 1778–1783.

Sacerdoti, E. D. 1974. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence* 5(2): 115–135.

Shanahan, M. 1989. Prediction Is Deduction but Explanation Is Abduction. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-89)*, 1055–1060.

Tate, A. 1977. Generating Project Networks. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence (IJCAI-77)*, 888–893. Cambridge, MA: Morgan Kaufmann.

Wilkins, D. E. 1984. Domain-Independent Planning: Representation and Plan Generation. *Artificial Intelligence* 22(3): 269–301.

Acknowledgements

This research was funded, in part, by the U.S. Government. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

Additional Contributions (AI algorithm and software development): Meng Ye, Abhinav Rajvanshi, Aravind Sundaresan, Aaron Spaulding, Phil Miller, Laura Tam, and Yi Yao.