Public Announcement Logic with Misinterpretations: Short Paper/Extended Abstract

Masayuki Tashiro[1,2][0000−0001−6511−1447], Eric Pacuit[1,3][0000−0002−0751−9011], and Ilaria Canavotto[1,4][0000−0002−8649−1810]

[1] University of Maryland, College Park, Maryland MD 20742, USA
[2] mtashiro@umd.edu
[3] epacuit@umd.edu
[4] icanavot@umd.edu

Abstract. An implicit Assumption in many dynamic logics of knowledge and belief is that all agents correctly interpret all of the formulas. However, it is not hard to imagine situations in which some agents misinterpret a formula. Such situations may arise because the agents have different awareness, because the language is ambiguous, because the agents are attending to different questions, or because the agents interpret the evidence differently. The aim of this paper is to develop a logic of public announcement allowing us to reason about the above mentioned situations and to study the dynamics underlying the correction of the agents' misinterpretation of a formula. In this framework, a public announcement of φ can thus be thought of as a public correction of each agent's interpretation of φ, followed by a simultaneous private announcement of φ to each agent.

Keywords: Public Announcement Logic · Awareness · Interrogative Epistemology · Ambiguous Language.

An implicit assumption in many dynamic logics of knowledge and belief is that all agents correctly interpret all of the formulas. However, it is not hard to imagine situations in which some agents *misinterpret* a formula. Such situations may arise because the agents have different awareness [7, 5, 6, 9], because the language is ambiguous [4], because the agents are attending to different questions [12, 1], or because the agents interpret the evidence differently [2]. When this happens, it is possible that the public announcement of a formula φ does not result in a state where there is common knowledge of φ (even if the formula announced is Boolean), since some of the agents may misinterpret the announcement of φ and believe that another formula φ' is true instead.

The aim of this paper is to develop a logic of public announcement allowing us to reason about the above mentioned situations and to study the dynamics underlying the *correction* of the agents' misinterpretation of a formula. To do this, we introduce new dynamic modal operators $[\varphi!]_i$ representing the event that agent i *corrects* her interpretation of φ. If each agent's interpretation of

φ is corrected before φ is announced, then a public announcement of φ results in everybody believing that φ.[5] In this framework, a public announcement of φ can thus be thought of as a *public correction* of each agent's interpretation of φ, followed by a simultaneous *private announcement* of φ to each agent.

In order to discuss the main features of our system, let us consider the following example.

Example 1 Two agents, Ann and Bob, are betting on a coin flip. They use a Japanese 10 yen coin. On the one hand, Ann, who is familiar with Japanese coins, correctly interprets the sentence that the coin lands heads (H) as H and the sentence that the coin lands tails (T) as T. On the other hand, Bob, who has never seen a Japanese coin before, misinterprets the sentences H and T, by interpreting H as T and T as H. After observing that the coin lands on heads, Ann believes that H is true, while Bob believes that T is true.

Our approach in modeling Ann's and Bob's interpretations of the sentences is *syntactical*: for each agent i, we define an *agent-relative interpretation function* λ_i that maps each formula to the formula on which it is interpreted. For instance, in Example 1, we have:

- $\lambda_{Ann}(H) = H$ and $\lambda_{Ann}(T) = T$, because Ann correctly interprets H and T;
- $\lambda_{Bob}(H) = T$ and $\lambda_{Bob}(T) = H$, because Bob misinterprets H and T.

An alternative, *semantic* approach is proposed by Halpern and Kets [4], who develop an Epistemic Probabilistic Logic featuring, for each agent i, an agent-relative *valuation* function that assigns to each formula a set of possible states—the proposition that agent i associates with that formula. So, going back to Example 1, according to [4], H is true for Ann but false for Bob, and T is false for Ann but true for Bob. The key difference with our proposal is that, in our framework, the truth value of a formula is determined by a single, agent-independent valuation function. So, in the above example, H is true at the actual state, even if Bob misinterprets H as $\lambda_{Bob}(H)$.

An advantage of our syntactic approach is that it allows us to represent not only the fact that Bob misinterprets H, but also *how* Bob misinterprets H (cf. [13, 8]). In general, there are three interesting cases:

1. $\lambda_{Bob}(H) = T$
2. $\lambda_{Bob}(H) = \mathsf{I}$, where I is a special formula for '*ignored*';
3. $\lambda_{Bob}(H) = \bot$,

Case 1 can be thought of as involving ambiguous evidence; the evidence that the coin landed heads is ambiguous in the sense that, while Ann interprets it as H, Bob interprets it as T. In this case, if H is publicly announced, Bob would come to believe that T is true rather than H.

[5] We restrict our attention to cases in which the announced formula φ is Boolean since there are formulas, such as Moorean sentences, that are never successful after they are publicly announced (cf. [14]).

Case 2 is analogous to situations studied in logics of question (see, e.g., [12, 1, 3]). In the Interrogative Logic, introduced by Baltag *et. al.* [12], agent i who does not currently attend the question 'whether φ?' ignores the public announcement of φ. In our system, this is represented by letting $\lambda_i(\varphi) = \mathsf{I}$. When a formula that i maps to I is publicly announced, we stipulate that i simply ignores the announcement and does not change her beliefs. Our use of the special formula I allows us to generate agent-relative sub models with the objective models by collapsing the set of states where the only difference is the truth of formula that the agent ignores. This is a commonly used strategy in modeling agents' unawareness [7, 9] and lack of inquisitive interest [12, 1, 3]. By using agent-relative interpretation function λ_i, we defined i's unawareness of φ as φ not being in the image of λ_i. This means that if i is unaware of φ, then there is no formulas (including φ itself) that i interprets as φ. In our running example, if $\lambda_{Bob}(H) = \mathsf{I}$, then Bob would not go through any belief updates even when H is publicly announced, so no change in beliefs occurs for Bob.

Finally, Case 3 is a situation where Bob considers a certain outcome of the coin flip to be impossible. This case reflects Savage's discussion about the large and small world problem where he argues that not taking a certain formula into account "may be useful in case certain states are regarded by [i] as virtually impossible so that they can be ignored" [10, p.9-10]. In this case, Bob would end up believing everything if H was publicly announced.

Regarding *corrections* of one's interpretation, there are three different ways to model this operation. Consider Bob's misinterpretation of H as $\lambda_i(H) = T$ (so Bob misinterprets Heads as Tails). The first way to correct Bob's misinterpretation is to make a public announcement of the form 'H if and only if T'. The second way is to change the valuation function so that the proposition expressed by H is assigned the proposition expressed by T (cf. dynamic epistemic logic with *assignment* [15]). We take a different approach in which correcting a misinterpretation of φ for agent i updates the agent-relative interpretation function so that $\lambda_i(\varphi) = \varphi$. For instance, correcting Bob's misinterpretation of H involves changing λ_{Bob} to λ'_{Bob} which is the same as λ_{Bob} except $\lambda'_{Bob}(H) = H$. This change to the agent-relative interpretation function is triggered by the correction modality $[\varphi!]_i$.

Our first contribution is to study how the correction modality $[\varphi!]_i$ interacts with the standard public announcement modality. Of course, correcting i's interpretation of φ followed by a public announcement of φ will be successful in the sense that i will believe φ (assuming that φ is Boolean). But, what is the effect on i's beliefs if φ is announced before i corrects her interpretation of φ? In some situations, correcting an interpretation of φ may trigger a belief change in which i may appear to *contract* her previously misinterpreted beliefs.

In the full version of the paper, we will explore how our syntactical approach is related to the alternative semantical approach, both in terms of modeling agents' misinterpretations and the correction of agents' misinterpretations. In addition, we will define an agent's *explicit beliefs* taking into account her misinterpretations. Typically, in the literature on unawareness, explicit belief is defined

in terms of the agent's implicit beliefs and an awareness operator. However, in many applications it is preferable to give a direct definition of the agents explicit beliefs (for instance, Schipper notes that "[i]n economics, we are only interested in knowledge that the agent is aware of, that can guide her decisions, and that in principle could be tested with choice experiments" [11, p. 10]). We give a direct definition of i's explicit beliefs: i believes what i interprets to be true in all states i considers possible with i's limited awareness of the situation. More formally, i explicitly believes φ at state w provided that there is some formula α such that $\lambda_i(\alpha) = \varphi$ and α is true at all states v that i considers possible given i's information and possible misinterpretations. This notion of possibility in light of i's possible misinterpretation is important. Consider, for example, a situation in which Bob is not aware that the coin could land on its edge (E). The state where E is true and both H and T false is impossible in light of what Bob is aware of as he considers only two outcomes to be possible (H or T), and in every state $H \vee T$ must be true. As such, some states may be implicitly accessible for Bob, but he may not consider them possible due to his limited awareness of the situation, and his explicit beliefs should only concern what are true at those implicitly accessible states that Bob considers possible.

References

1. Baltag, A., Boddy, R., Smets, S.: Group knowledge in interrogative epistemology. In: Ditmarsch, H.V., Sandu, G. (eds.) Jaakko Hintikka on Knowledge and Game Theoretical Semantics. Springer (2018)
2. Benthem, J., Ştefan Minică: Toward a dynamic logic of questions. Journal of Philosophical Logic **41**(4), 633–669 (2012)
3. Bjorndahl, A., Özgün, A.: Uncertainty about evidence. In: Moss, L.S. (ed.) Proceedings Seventeenth Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2019, Toulouse, France, 17-19 July 2019. EPTCS, vol. 297, pp. 68–81 (2019). https://doi.org/10.4204/EPTCS.297.5, https://doi.org/10.4204/EPTCS.297.5
4. Ciardelli, I., Groenendijk, J., Roelofsen, F.: Inquisitive Semantics. Oxford University Press (2018)
5. Halpern, J.Y., Kets, W.: Ambiguous language and differences in beliefs (2012). https://doi.org/10.48550/ARXIV.1203.0699, https://arxiv.org/abs/1203.0699
6. Halpern, J.Y., Rêgo, L.: Interactive unawareness revisited. Games and Economic Behavior **62**, 232 – 262 (2008)
7. Halpern, J.Y., Rêgo, L.: Reasoning about knowledge of unawareness revisited. Mathematical Social Sciences **65**, 73 – 84 (2013)
8. Heifetz, A., Meier, M., Schipper, B.: Interactive unawareness. Journal of Economic Theory **130**(1), 78–94 (2006), https://EconPapers.repec.org/RePEc:eee:jetheo:v:130:y:2006:i:1:p:78-94
9. Mahtani, A.: Awareness growth and dispositional attitudes. Synthese **198**, 8981–8997 (2021). https://doi.org/10.1007/s11229-020-02611-5
10. Piermont, E.: Algebraic semantics for relative truth, awareness, and possibility (2021)
11. Savage, L.J.: The Foundations of Statistics. Wiley Publications in Statistics (1954)

12. Schipper, B.: Awareness. In: Handbook of Epistemic Logic. pp. 77 – 146 (2014)
13. Steele, K., Stefánsson, H.O.: Beyond Uncertainty: Reasoning with Unknown Possibilities. Cambridge University Press (2021)
14. van Ditmarsch, H., Kooi, B.: The secret of my success. Synthese **151**(2), 201 – 232 (2006). https://doi.org/10.1007/s11229-005-3384-9
15. van Ditmarsch, H., van der Hoek, W., Kooi, B.: Dynamic epistemic logic with assignment. In: AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems. pp. 141 – 148 (2005). https://doi.org/10.1145/1082473.1082495

Dynamic Logic for Descriptive Complexity*

Eugenia Ternovska

Simon Fraser University, Canada, ter@sfu.ca

Abstract. The central open question in Descriptive Complexity is whether there is a logic that characterizes deterministic polynomial time (P-time) on relational structures. Towards this goal, we define a logic that is obtained from first-order logic with fixed points, FO(FP), by a series of transformations that include restricting logical connectives. The formalism can be viewed, simultaneously, as an algebra of binary relations and as a modal Dynamic Logic, where algebraic expressions describing "proofs" or "programs" appear inside the modalities. The logic can express both reachability and counting properties on unordered structures, and can encode an arbitrary P-time Turing machine. The data complexity of model checking for the logic is in non-deterministic polynomial time (NP). A crucial question is under what syntactic conditions on the algebraic terms checking just one certificate for the membership in NP is sufficient. This paper sets mathematical foundations for such a study.

1 Introduction

The goal of Descriptive Complexity is to characterize computational complexity in a machine-independent way, in order to apply logical and model-theoretic methods to the study of complexity [23]. However, this goal presents multiple challenges. A logical characterization of NP was given by Fagin. His celebrated theorem [13] states that the complexity class NP coincides, in a precise sense, with second-order existential logic. The central open question in the area is whether there is a logic that exactly characterizes deterministic polynomial-time (P-time) computability on relational structures. The problem was first formulated by Chandra and Harel [8] and made precise by Gurevich [20] (see also [19]). Following the formulations of Dawar [9], we have:

Definition 1. *A logic L is a function* SEN *associating a recursive set of sentences to each finite vocabulary τ together with a function* SAT *that associates to each τ a recursive satisfaction relation relating finite τ-structures to sentences that is also isomorphism-invariant. That is, if $\mathfrak{A}$ and $\mathfrak{B}$ are isomorphic τ-structures and ϕ is any sentence of $\mathsf{SEN}(\tau)$ then $(\mathfrak{A}, \phi) \in \mathsf{SAT}(\tau)$ if, and only if, $(\mathfrak{B}, \phi) \in \mathsf{SAT}(\tau)$.*

Definition 2. *A logic L* captures P-time *if*

(i) there is a computable function that takes each sentence of L to a polynomial time Turing machine that recognises the models of the sentence, and

(ii) for every P-time *recognizable class $\mathcal{K}$ of structures, there is a sentence of L whose models are exactly $\mathcal{K}$.*

* Supported by Natural Sciences and Engineering Research Council of Canada (NSERC).

Currently, P-time is known to be captured on *linearly ordered* structures only. On such structures, first-order logic with fixed points, FO(FP), captures the complexity class P-time. The result was independently obtained by Immerman and Vardi in the 1980's [22, 32]. However, even extended with counting terms, FO(FP) + C, the fixed-point logic is too weak to express all polynomial-time properties on structures without such order, as shown by Cai, Fürer and Immerman in [7]. For *arbitrary* structures, capturing P-time remains an intriguing open problem, that, so far, has resisted any attempt to solve it. Nevertheless, there has been a lot of development in the area. Dawar, Grohe, Holm and Laubner introduced Rank logic in [10]. More recent work includes that by Pakusa, Schalthöfer and Selman [27], Grädel and Pakusa [17], Atserias, Dawar and Ochremiak [2], Dawar and Wilsenach [11], among others. The recent result by Lichter separated Rank logic [10] from P-time [25]. Despite the resent advances, Choiceless Polynomial Time [4] proposed in 1999, remains perhaps the most prominent candidate for a logic for P-time. In a related area of Constraint Satisfaction Problem (CSP), a much more significant progress has been made. The long-standing Fedor-Vardi dichotomy conjecture has been resolved by Bulatov and Zhuk in 2017 [6, 35]. Universal-algebraic techniques played a pivotal role in that development. Unfortunately, the current approaches in Descriptive Complexity do not allow for such techniques. In particular, explicit counting constructs, that are used in several approaches to the problem, e.g., [5, 10, 18] are an obstacle toward algebraization.

In this paper, we develop an *algebraic* view on the problem. Our algebra is, simultaneously, a modal Dynamic Logic, where programs (or proofs) appear inside the modalities. This approach is very different from any earlier attempts to characterize P-time. Our strategy is to allow the logic express *exactly* NP, and then use techniques of Universal Algebra to look for syntactic conditions for P-time.

To obtain the logic, we started with FO(FP), the logic for which a partial solution to the main problem is known by the Immerman-Vardi theorem, [22, 32], and specified inputs and outputs of atomic expressions. We obtained an algebra of binary relations, similar to Jonsson and Tarski [24]. The step towards binary relations was inspired, in part, by the good properties of two-variable fragments of FO [31] that have, e.g., order-invariance [34]. Unfortunately, such fragments are not expressive enough to encode Turing machines. To overcome this obstacle, we lifted the algebra to operate on *binary relations on relational structures*. Such relations, intuitively, encode state transitions. That initial algebra [30], based on FO(FP), became our first tool to analyze how information propagates from inputs to outputs. We then investigated what syntactic features make information flows efficient. We identified two such parameters: the algebraic operations and the logic that specifies atomic units (here, called *modules*). The atomic units were taken to be a modification of conjunctive queries. Intuitively, they represent conditional non-deterministic assignments. We took our algebraic operations to be a *restriction of FO connectives*, similarly to Description logics [3]. Restricting negation (full complementation) to its unary counterpart is already known to imply good computational properties [28].[1] However, in our case, *all* connectives and the iterator construct of FO(FP) had to be restricted. In particular, we replaced the "static" (Boolean) con-

[1] Unary negation is related to the negation of modal logics. Such logics are known to be robustly decidable [33, 16].

junction and disjunction with their "dynamic" counterparts – sequential composition and preferential union. Restricting FO connectives allowed us to avoid some known undecidability pitfalls. Despite its simplicity, the algebra defines the main constructs of imperative programming, such as `if-then-else` and `while` loops. Thus, each algebraic term can be thought of as a program, where atomic operations are conditional non-deterministic assignments. To control the complexity of the computation, rich tests (i.e., those extended over time) are limited to querying the trace of the program. Moreover, all relations that change from state to state are singleton-set monadic. This restriction allows us to control the length of the computation entirely by logical means, unlike the externally imposed polynomial bounds in [4]. Many typical P-time properties on unordered structures such as several cardinality properties (e.g., the query EVEN asking if the size of the input domain is even), reachability-type properties, and those requiring "mixed" propagations, e.g., mod 2 linear equations (that include the CFI example [7], see [1]), are also expressible. In this paper, we show that: (1) The logic encodes an arbitrary P-time Turing machine by guessing a linear order on domain elements first. (2) The data complexity of model checking is in NP, and one might need to guess potentially exponentially many traces. The remaining goal is to find under what conditions on the algebraic terms, the traces are indistinguishable in a precise mathematical sense, so that checking one (any) of them is enough. Decidability of these symmetry conditions would give us a logic for P-time – the syntax would include the logic presented here, plus these algebraic conditions. Both Definitions 1 and 2 would be satisfied. The main contribution of this paper is to set the stage for an algebraic study of such symmetries.

The rest of the paper is organized as follows. In **Section 2**, we introduce basic notations and give technical preliminaries. In **Section 3**, we present the syntax of the algebra and the atomic transitions, and in **Sections 4 and 5**, we define the semantics in terms of binary relations on relational structures. In **Section 6**, we reformulate the algebra as a modal Dynamic Logic and show how to define the main programming constructs. In **Section 7**, we define the notion of a computational problem specified by an algebraic term, and the set of certificates for such a problem. **Section 8** shows how any P-time Turing machine can be encoded. We conclude, in **Section 9**, with a summary.

2 Technical Preliminaries

In this section, we set basic notations and terminology. We review Conjunctive Queries and PP-definable relations. Importantly, we introduce a Choice construct that, given a set, selects an element of it. We assume familiarity with the basic notions of first-order (FO) and second-order (SO) logic (see, e.g., [12]) and use ':=' to mean "is by definition".

Conjunctive Queries and SM-PP-Definable Relations Let τ be a relational vocabulary, which is finite (but is of an unlimited size). Let $\tau := \{S_1, \ldots, S_n\}$, each S_i has an associated arity r_i, and $\mathcal{D}$ be a non-empty set. A *τ-structure* $\mathfrak{A}$ over domain $\mathcal{D}$ is $\mathfrak{A} := (\mathcal{D};\ S_1^{\mathfrak{A}}, \ldots, S_n^{\mathfrak{A}})$, where $S_i^{\mathfrak{A}}$ is an r_i-ary relation called the *interpretation* of S_i. In this paper, all structures are finite. If $\mathfrak{A}$ is a τ-structure, $\mathfrak{A}|_\sigma$ is its restriction to a sub-vocabulary σ. Let $\mathbf{U}$ denotes the set of all τ-structures over the same domain $\mathcal{D}$.

Let C be a class of FO structures of some vocabulary τ. Following Gurevich [20], we say that an r-ary C-*global relation* Q (a *query*) assigns to each structure $\mathfrak{A}$ in C an r-ary relation $Q(\mathfrak{A})$ on $\mathfrak{A}$; the relation $Q(\mathfrak{A})$ is the specialization of Q to $\mathfrak{A}$. The vocabulary τ is the vocabulary of Q. If C is the class of all τ-structures, we say that Q is τ-*global*. Let $\mathcal{L}$ be a logic. A k-ary query Q on C is $\mathcal{L}$-definable if there is an $\mathcal{L}$-formula $\psi(x_1,\ldots,x_k)$ with $x_1,\ldots,x_k$ as free variables such that for every $\mathfrak{A} \in C$, $Q(\mathfrak{A}) = \{(a_1,\ldots,a_k) \in \mathcal{D}^k \mid \mathfrak{A} \models \psi(a_1,\ldots,a_k)\}$. Query Q is *monadic* if $k = 1$. A *conjunctive query* (CQ) is a query definable by a FO formula in prenex normal form built from atomic formulas, $\wedge$, and $\exists$ only. A relation is *Primitive Positive (PP) definable* if it is definable by a CQ: $\forall x_1 \ldots \forall x_k \big(R(x_1,\ldots,x_k) \leftrightarrow \exists z_1 \ldots \exists z_m \, (B_1 \wedge \cdots \wedge B_m)\big)$, and each B_i has object variables from $x_1,\ldots,x_k,z_1,\ldots,z_m$.

Example 1. Relation **Path of Length Two** is PP-definable, but not monadic:
$$\forall x_1 \forall x_2 \, \big(Z(x_1,x_2) \leftrightarrow \exists z (Y(x_1,z) \wedge Y(z,x_2))\big).$$

Example 2. Relation **At Distance Two from** X is monadic and PP-definable:
$$\forall x_2 \, \big(Z(x_2) \leftrightarrow \exists x_1 \exists z \, (X(x_1) \wedge Y(x_1,z) \wedge Y(z,x_2))\big).$$

We will restrict relations as in Example 2 to put just one (arbitrary) element in their extension.

Definition 3. *Singleton-set-Monadic Primitive Positive (*SM-PP*) relation* is a singleton-set relation R implicitly definable by:

$$\forall x \, \big(R(x) \rightarrow \big(\underbrace{\exists z_1 \ldots \exists z_m \, (B_1 \wedge \cdots \wedge B_m)}_{\text{Conjunctive Query}} \wedge \forall x \forall y (R(x) \wedge R(y) \rightarrow x = y)\big)\big), \tag{1}$$

where each B_i is a relational variable from $\mathbb{V}$ with object variables from $x, z_1, \ldots, z_m$ such that it is either (a) unary or (b) the interpretation of the vocabulary symbol $s(B_i)$ comes from the input structure $\mathfrak{A}$ (i.e., it is EDB in the database terminology). We use a rule-based notation for (1):

$$R(x) \leftharpoondown B_1, \ldots, B_m. \tag{2}$$

Notation $\leftharpoondown$, unlike Datalog's $\leftarrow$, is used to emphasize that **only one element is put into the relation in the head of the rule** (i.e., the relation on the left of $\leftharpoondown$).

Example 3. Suppose we want to put just one (arbitrary) element in extension of the relation Z denoting At Distance Two from Example 2. In the rule-based syntax (2):

$$\big\{\, Z(x_2) \leftharpoondown X(x_1), Y(x_1,z), Y(z,x_2)) \,\big\}.$$

The defined relation is SM-PP. Since only one domain element, out of those at distance two from X, is put into Z, there could be multiple outcomes, depending on the choice.

Example 4. A non-example of SM-PP relation is Path of Length Two (Example 1). The relation on the output is neither monadic nor singleton-set.

Strings and Trees Let Σ be an alphabet. A *finite word* (or a *string*) is a finite sequence $w := a_0 \cdot a_1 \dots a_n$ of letters in Σ, where the indexes are called *positions*. We have a binary relation on words $w \sqsubseteq w'$ to denote that w is a prefix of w'. We use notation w_i for the prefix of w ending in position $i \in \mathbb{N}$. The *length* of w is $|w| := n + 1$. The empty word, i.e., such that $|w| = 0$, is denoted $\mathbf{e}$. The *positions* in word w are denoted as $i, j \in \mathbb{N}$. We use $w(i)$ for the i-th letter in word w. Let $last(w) := |w| - 1$. We write $w(last)$ to denote $w(last(w))$. A word such that $w(0) = a$, where $a \in \Sigma$, is denoted $w(a)$. Later in the paper, Σ will be $\mathbf{U}$, the set of all τ-structures over the same domain.A *tree* over Σ is a (finite or infinite) nonempty set $Tr \subseteq \Sigma^*$ such that for all $x \cdot c \in Tr$, with $x \in \Sigma^*$ and $c \in \Sigma$, we have $x \in Tr$. The elements of Tr are called *nodes*, and the empty word $\mathbf{e}$ is the *root* of Tr. For every $x \in Tr$, the nodes $x \cdot c \in Tr$ where $c \in \Sigma$ are the *children* of x. A node with no children is a *leaf*. We refer to the length $|x|$ of x as its *level* in the tree.

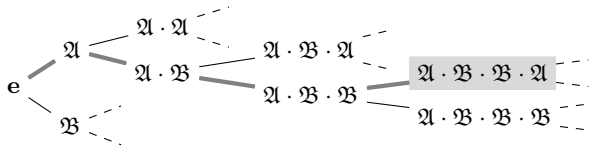

Fig. 1. A Full tree over $\mathbf{U}$. Some of its nodes (such as the shaded one) may correspond to traces of programs (or proofs).

A $|\Sigma|$-ary tree is *full* if each node is either a leaf or has exactly $|\Sigma|$ child nodes. It is *Full* if it has no leaves. A *branch* b of a tree Tr is a set $b \subseteq Tr$ such that $\mathbf{e} \in b$ and for every $x \in b$, either x is a leaf, or there exists a unique $c \in \Sigma$ such that $x \cdot c \in b$. In a Full tree, every branch is infinite.

3 Algebra of Binary Relations: Syntax

In this section, we introduce the syntax of an algebra of binary relations on strings. Intuitively, well-formed expressions in this algebra represent programs.

A *(module) vocabulary* $\mathcal{M}$ is a triple $(Names, \mathrm{ar}, \mathrm{iar})$:

- $Names$ is a nonempty set, the elements of which are called *module names*;
- ar assigns an arity to each module name in $Names$;
- iar assigns an input arity to each module name M in $Names$, where $\mathrm{iar}(M) \leq \mathrm{ar}(M)$.

In this paper, each module name in $Names$ will refer to a set of rules of the form (2). We also fix a countably infinite supply of relational variables $\mathbb{V}$. Our algebraic expressions will use variables in $\mathbb{V}$. A function $s : \mathbb{V} \to \tau$ (discussed in the Semantics section) will instantiate variables in $\mathbb{V}$ with the elements of a relational vocabulary τ.

Algebraic Syntax We call this syntax *one-sorted* to contrast it with the two-sorted one in the form of a modal Dynamic Logic in Section 6. Algebraic expressions (terms) are given by the grammar:

$$\alpha ::= \mathrm{id} \mid M(\bar{Z}) \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha \,;\, \alpha \mid \alpha \sqcup \alpha \mid X = Y \mid X =_t Y \mid \alpha^{\uparrow}, \qquad (3)$$

where $X =_t Y$ must occur within the scope of $\curvearrowright$ or $\curvearrowleft$. Here, M is any module name in $\mathcal{M}$, $\bar{Z}$ is a tuple of variables; and X, Y are variables in $\mathbb{V}$. For *atomic module expressions*, i.e., expressions of the form $M(\bar{Z})$, the length of $\bar{Z}$ must be equal

to $\mathrm{ar}(M)$. In practice, we will often write $M(\bar{X};\bar{Y})$ for atomic module expressions, where $\bar{X}$ is a tuple of variables of length $\mathrm{iar}(M)$ and $\bar{Y}$ is a tuple of variables of length $\mathrm{ar}(M) - \mathrm{iar}(M)$. In addition, we have: Identity (Diagonal) (id), atomic module symbols ($M(\bar{Z})$), Sequential Composition (;), Forward-facing ($\curvearrowright$) and Backwards-facing ($\curvearrowleft$) Unary Negations, also called Anti-Domain and Anti-Image, respectively, Preferential Union ($\sqcup$), which is a restriction of Union, Equality Test ($X = Y$), Comparison over Time ($X =_t Y$), and Maximum Iterate ($^{\uparrow}$), which is a modification of transitive closure that outputs the longest transition.

Atomic Transitions: SM-PP-Definable Our algebra has a **two-level syntax**: in addition to the algebraic expressions (3), atomic state-to-state transitions are as follows.

Definition 4. *An* SM-PP transition (atomic module) *is a set of rules of the form (2).*

Thus, SM-PP transitions have: **Outputs:** SM-PP definable relations, as per Definition 3, and **Inputs:** monadic or given by the input structure $\mathfrak{A}$.
Atomic modules provide a transition system context, denoted T, for the algebra. In this context, the only relations that change from structure to structure are unary and contain one domain element at a time, similarly to registers in a Register Machine [29].

4 Semantics of Atomic State Transitions

Next, we provide a meaning to atomic transitions.We give an example of how algebraic expressions are used, and provide some intuitions about the underlying machine model.

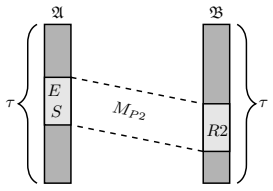

Fig. 2. State transition.

To gain intuitions, consider a module with the name M_{P2}. Let $\mathfrak{A}$ and $\mathfrak{B}$ be relational τ-structures. In each transition $(\mathfrak{A}, \mathfrak{B})$, the inputs of this module (E and S in Figure 2) are "read" in $\mathfrak{A}$ and the outputs ($R2$ in Figure 2) are updated in $\mathfrak{B}$. Relations unaffected by the update are copied from $\mathfrak{A}$ to $\mathfrak{B}$. This preservation principle for atomic transitions, here formalized by (4), appeared as a commonsense law of inertia [26].

Following these intuitions, we specify a transition system $T(M)$ in (4). For each M, it is a binary relation on τ-structures, as in Figure 2. The reader can skip the details at the first reading, and go directly to Example 6 (a specification of EVEN).

A **semantic context** $T := T(s, \mathcal{D}, \mathcal{I}(\cdot))$ is a function that maps each atomic symbol M in $\mathcal{M}$ to a binary relation on the set $\mathbf{U}$ of all τ-structures over the domain $\mathcal{D}$. This, a semantic context T gives a transition relation on $\mathbf{U}$. Function T is parameterized with:

- a variable assignment (an instantiation function) $s : \mathbb{V} \to \tau$, where τ is a relational vocabulary;
- a base set $\mathcal{D}$, which is, in this paper, the domain of the input structure;
- a *static* interpretation $\mathcal{I}(M)$ of module names M in $\mathcal{M}$. For each atomic module name M in $\mathcal{M}$ and tuples of variables $\bar{Z}$ and $\bar{X} \subseteq \bar{Z}$, with $|\bar{Z}| = \mathrm{ar}(M)$ and $|\bar{X}| = \mathrm{iar}(M)$, static interpretation $\mathcal{I}(M)$ is a function $\mathcal{I}$ that returns a set of $s(\bar{Z})$-structures ($s(\bar{Z}) \subseteq \tau$) with domain $\mathcal{D}$, where an *interpretation of* $s(\bar{X})$ *is expanded*

to the entire vocabulary $s(\bar{Z})$ to satisfy a specification in a logic $\mathcal{L}_T$.[2] Here, $\mathcal{L}_T$ is given by Definition 4.

To finish defining semantic context $T := T(s, \mathcal{D}, \mathcal{I}(\cdot))$, we reinterpret each static interpretation dynamically, that is, as a **state transition** represented by a **dynamic relation (DR)**, i.e., a binary relation on τ-structures, as in Figure 2. For *atomic* DRs, we require that, relations, that are not explicitly modified by the expansion from $s(\bar{X})$ to $s(\bar{Z})$, remain the same. This includes interpretations of unconstrained variables that are interpreted arbitrarily. Taking into account this intuition, we now present atomic DRs formally. Static interpretation $\mathcal{I}(M)$, for each atomic module name M, gives rise to a **binary relation** $T(M)$ on $\mathbf{U}$ defined as follows:

$$\begin{array}{c} T(M(\underline{\bar{X}};\bar{Y})) := \big\{(\mathfrak{A},\mathfrak{B}) \in \mathbf{U}\times\mathbf{U} \mid \text{ exists } \mathfrak{C} \in \mathcal{I}(M) \text{ and} \\ \mathfrak{C}|_{s(\bar{X})} = \mathfrak{A}|_{s(\bar{X})}, \quad \mathfrak{C}|_{s(\bar{Y})} = \mathfrak{B}|_{s(\bar{Y})}, \quad \text{and} \quad \mathfrak{A}|_{\tau - s(\bar{Y})} = \mathfrak{B}|_{\tau - s(\bar{Y})}\big\}. \end{array} \tag{4}$$

Example 5. Consider, again, Figure 2. Let M_{R2} in module vocabulary $\mathcal{M}$ be the name of the atomic module from Example 3 specifying At Distance Two. Let τ contain predicate symbols E (edge), S (source) and $R2$ (reach-in-two-steps), among other symbols, and let the instantiation function s maps relational variables X, Y and Z to S, E and $R2$, respectively. Then static interpretation $\mathcal{I}(M_{R2})$ is a set of all $s(X,Y,Z)$-structures over some domain $\mathcal{D}$ that are the models of the expression from Example 3. Now, suppose, according to the vocabulary $\mathcal{M}$, we have: $M_{R2}(\underline{X},\underline{Y},Z)$, where the variables X and Y in the input positions are underlined. Then by (4), we have a binary relation on τ-structures, with $s(X,Y,Z) \subseteq \tau$. Edges $E = s(Y)$ and the start node $S = s(X)$ are "read" in a τ-structure $\mathfrak{A}$ on the left, and a node reachable in two steps (just one of such nodes) is put in $R2 = s(Z)$ in $\mathfrak{B}$ on the right.

Example 6. Before we define the semantics of the algebraic operations, we show how they will be used. We consider the problem EVEN, that, given a structure $\mathfrak{A}$ with an empty vocabulary, checks: Is $|dom(\mathfrak{A})|$ even? EVEN is P-time computable, but is not expressible in MSO, Datalog, or any fixed point logic, *unless* a linear order on the domain elements is given. We construct a path in the transition system in two-step increments. Here, again, we underline the variables in $\mathbb{V}$ that appear in the input positions. We also use more informative names for the relational variables, which will be our practice from now on. We use a secondary numeric domain with successor function $succ$. The minimal and maximal elements of this domain are stored in relations *Min* and *Max*, respectively, and Max is the cardinality of the main domain. Module *InitiateCounter* $:= \big\{ C(x) \leftarrowtail \underline{Min(x)} \big\}$ initiates the counter to the minimal element stored in the relation Min. Action *IncreaseCounter* $:= \big\{ C(succ(x)) \leftarrowtail C(x)) \big\}$ increases the counter. Transition *GuessP* $:= \big\{ P(x) \leftarrowtail \big\}$ puts all possible domain elements into P and chooses one of them. We define a subprogram:

$$\textit{GuessNew} := \textit{GuessP} \,;\, \textit{NewP}? \,;\, \textit{IncreaseCounter}.$$

The problem EVEN is now specified as:

$$\alpha_E := \mathbf{repeat}\ (\textit{GuessNew} \,;\, \textit{GuessNew})\ \mathbf{until}\ C = \textit{Max}.$$

[2] We abuse the notations slightly by writing $s(\bar{Z})$ to denote the set $\{s(Z_1), s(Z_2)\dots\}$, where $Z_1, Z_2, \dots$ are variables in $\bar{Z}$.

The rich test *NewP*? within *GuessNew* is represented by $\neg\langle\, P{=}_t P \mid \mathbf{T}?$, where $\langle\cdots\mid$ denotes a left-facing existential modality evaluated *within the trace of the program*, $\mathbf{T}$ denotes *True*, and $P{=}_t P$ is a binary relation that returns pairs of situations where the current value stored in P equals the one in the past. The rich test, evaluated now, checks if there is no situation in the past where the value stored in P is the same as now. The modality and the Repeat-Until construct are *definable* in the algebra (3).

Given a structure $\mathfrak{A}$ over an empty vocabulary, α_E is *executed successfully* (a term formally defined in Section 7) if and only if the size of the input domain is even.

Machine Model We can think of evaluations of algebraic expressions such as α_E as *computations of Register machines* (see, e.g., [29]) . We have: (1) monadic "registers" – predicates used during the computation, each containing only one element at a time, such as P, C above; (2) the "real" inputs, such as edges $E(x, y)$ of an input graph, are of any arity; (3) atomic transitions correspond to conditional assignments with a non-deterministic outcome; (4) in each atomic transition, only the registers of the current state or the input structure $\mathfrak{A}$ are accessible; (5) registers can be checked for equalities only and no elements are referred to directly (i.e., via constants); (6) computations are controlled by algebraic terms that, intuitively, represent programs.

While, in each state, the registers are singletons, their content is accumulated in the guessed path (a finite sequence of τ-structures). Moreover, in Example 6, each path gives us an implicit linear order on the guessed domain elements.

5 Semantics of Algebraic Operations

We now introduce a semantics of the algebraic operations as binary relations on *strings* of relational τ-structures. That is, (3) is an *algebra of transductions* or string-rewriting operations.

Global Dynamic Relations Recall that DR stands for a Dynamic Relation. We, again, adapt Gurevich's terminology [20], as we did in Section 2. We say that every algebraic expression α denotes a *global DR* $[\![\alpha]\!]$: a function that maps T to the DR $\alpha(T) := [\![\alpha]\!]_T$. Next, we give the semantics of the main operations as binary relations.

Identity (Diagonal) $[\![\mathrm{id}]\!]_T := \{(w, w') \in \mathbf{U}^+{\times}\mathbf{U}^+ \mid w = w'\}$.

Atomic Transductions

$$[\![M(\underline{\bar{X}}; \bar{Y})]\!]_T := \big\{(w, w') \mid (w(last), w'(last)) \in T(M(\underline{\bar{X}}; \bar{Y})) \wedge w \sqsubseteq w'\big\}.$$

Sequential Composition

$[\![\alpha\,;\beta]\!]_T := \{(w, w') \in \mathbf{U}^+{\times}\mathbf{U}^+ \mid \exists w''\ ((w, w'') \in [\![\alpha]\!]_T$ and $(w'', w') \in [\![\beta]\!]_T)\}$.

Equality Test $[\![X = Y]\!]_T := \{(w, w) \in \mathbf{U}^+{\times}\mathbf{U}^+ \mid ((s(X))^{w(last)} = (s(Y))^{w(last)}\}$. That is, the interpretations of the two symbols in the last structure of string w coincide.

Maximum Iterate is a determinization of the Kleene star α^* (reflexive transitive closure). It outputs only the longest, in terms of the number of α-steps, transition out of all possible transitions produced by the Kleene star. We define it as: $\alpha^{\uparrow} = \bigcup_{0\le n<\omega} max(\alpha_n)$, where $\alpha_0 := \mathrm{id}$, and $\alpha_{n+1} := \alpha;\alpha_n$. Here, $max(\alpha_n)$ is an abbreviation that, intuitively,

means that applying α again ($n+1$'st time) is undefined. We omit the technical details of the formalization for lack of space.

Preferential union

$$[\![\alpha \sqcup \beta]\!]_T := \left\{(w,w') \in \mathbf{U}^+\times\mathbf{U}^+ \;\middle|\; \begin{array}{l}(w,w') \in [\![\alpha]\!]_T \text{ if } (w,w') \in [\![\alpha]\!]_T, \\ (w,w') \in [\![\beta]\!]_T \text{ if } (w,w') \notin [\![\alpha]\!]_T \\ \qquad\qquad \text{and } (w,w') \in [\![\beta]\!]_T\end{array}\right\}.$$

Thus, we perform α if it is defined, otherwise we perform β. This operation is a determinization of Union.

The meanings of $\curvearrowright$ and $\curvearrowleft$ are reduced to interpretations $[\![\cdot]\!]^w_T$ with respect to a trace w.

$[\![\curvearrowright \alpha]\!]_T := \{(w,w) \in \mathbf{U}^+\times\mathbf{U}^+ \mid (w,w) \in [\![\curvearrowright \alpha]\!]^w_T\}$.

$[\![\curvearrowleft \alpha]\!]_T := \{(w,w) \in \mathbf{U}^+\times\mathbf{U}^+ \mid (w,w) \in [\![\curvearrowleft \alpha]\!]^w_T\}$.

In-Trace Forward Unary Negation (Anti-Domain) says "from w_t, there is no outgoing α-transition that ends somewhere within the trace w_n".[3] The subscripts n and t in w_n and w_t intuitively stand for "now" and "then", respectively.

$$[\![\curvearrowright \alpha]\!]^{w_n}_T := \{(w_t,w_t) \in \mathbf{U}^+\times\mathbf{U}^+ \mid\; w_t \sqsubseteq w_n \wedge \forall w' \sqsubseteq w_n \;\; (w_t,w') \notin [\![\alpha]\!]^{w_n}_T\}.$$

In-Trace Backwards Unary Negation (Anti-Image) says "in w_t, there is no incoming α-transition that starts within the trace w_n". Thus,

$$[\![\curvearrowleft \alpha]\!]^{w_n}_T := \{(w_t,w_t) \in \mathbf{U}^+\times\mathbf{U}^+ \mid\; w_t \sqsubseteq w_n \wedge \forall w' \sqsubseteq w_t \;\; (w',w_t) \notin [\![\alpha]\!]^{w_n}_T\}.$$

Each of the unary negations is a restriction of the regular negation (complementation). Immediately from the semantics, we see that $\curvearrowright$ and $\curvearrowleft$ are not idempotent, $\curvearrowright\curvearrowright \alpha \neq \alpha$, but are weakly idempotent, $\curvearrowright\curvearrowright\curvearrowright \alpha = \curvearrowright \alpha$, as is the Intuitionistic negation. This is because Anti-Domain applied twice ($\curvearrowright\curvearrowright$) produces Domain, and the Domain of an Anti-Domain ($\curvearrowright\curvearrowright\curvearrowright$), is, again, an Anti-Domain ($\curvearrowright$) of α.

Comparison Extended over Time only appears in the scope of $\curvearrowright$ or $\curvearrowleft$, so is evaluated in a trace only. $[\![X{=_t}Y]\!]^{w_n}_T :=$
$\{(w,w') \in \mathbf{U}^+\times\mathbf{U}^+ \mid (s(X))^{w(last)} = (s(Y))^{w'(last)} \wedge w \sqsubset w' \sqsubseteq w_n\}$. That is, the operation returns pairs of strings (the second one extends the first) such that the interpretations of X and Y, in the last states of both strings, coincide.

In-Trace Other Operations id, M, ;, $\sqcup$, $^\uparrow$. Since other operations can occur withing a rich test, they are evaluated with respect to a trace as well. Their semantics $[\![\cdot]\!]^{w_n}_T$ is the same as $[\![\cdot]\!]_T$ above, except a condition is added that all strings are substrings of w_n.

6 Dynamic Logic

In this section, we provide an alternative (and equivalent) two-sorted version of the syntax (3) in the form of a Dynamic logic. While the two formalizations are equivalent, in many ways, it is easier to work with the logic. We demonstrate that the logic defines

[3] Ideally, these operations should be denoted with a notation closer to the typical negation sign, i.e., more "square", however we could not find a suitable pair of Latex symbols. Any suggestions for a better solution are welcome.

all major constructs of imperative programming. The syntax is given by the grammar:

$$\begin{array}{c} \alpha \ ::= \ \mathrm{id} \mid M_a(\bar{Z}) \mid \curvearrowright \alpha \mid \curvearrowleft \alpha \mid \alpha \,;\, \alpha \mid \alpha \sqcup \alpha \mid X = Y \mid X =_t Y \mid \alpha^{\uparrow} \mid \phi? \\ \phi ::= \mathbf{T} \mid M_p(\bar{Z}) \mid \neg\phi \mid |\alpha\rangle\ \phi \mid \langle\alpha|\ \phi. \end{array} \tag{5}$$

The subscripts a and p of M stand for "actions" and "propositions", respectively. Intuitively, proposition modules M_p make only self-loop transitions. The expressions in the first line of (5) are typically called *process* terms, and those in the second line *state* formulae. State formulae ϕ are "unary" in the same sense as $P(x)$ is a unary notation for $P(x,x)$. Semantically, they are subsets of the identity relation on $\mathbf{U}$.

The state formulae in the second line of (5) are shorthands that use the operations in the first line: $\mathbf{T} := \mathrm{id}$, $\neg\phi :=\curvearrowright\ \phi$, $|\alpha\rangle\ \phi :=\curvearrowright\curvearrowright\ \alpha\,;\,\phi$, $\langle\alpha|\ \phi :=\curvearrowleft\curvearrowleft\ \phi\,;\,\alpha$, and $\phi? :=\curvearrowright\curvearrowright\ \phi$. State formulae $|\alpha\rangle\ \phi$ and $\langle\alpha|\ \phi$ are right- and left-facing existential modalities, where, intuitively, the existential quantifier ranges over states in a path. We define $\mathrm{Dom}(\alpha) := \curvearrowright\curvearrowright\alpha$ and $\mathrm{Img}(\alpha) := \curvearrowleft\curvearrowleft\alpha$. With this notation, we have that $|\alpha\rangle\ \phi \ = \ \mathrm{Dom}(\alpha\,;\,\phi)$, and $\langle\alpha|\ \phi \ = \ \mathrm{Img}(\phi\,;\,\alpha)$. Clearly, $\mathrm{Dom}(\mathrm{Img}(\alpha)) = \mathrm{Img}(\alpha)$. We define universal modalities $|\alpha]\ \phi := \curvearrowright |\alpha\rangle\ \curvearrowright \phi$ and $[\alpha|\ \phi := \curvearrowleft \langle\alpha|\ \curvearrowleft \phi$ dually.

Example 7. We now give an example of evaluating formula $\langle M|(Q = R)?$. Intuitively, the semantics should return a set of pairs of identical strings (w, w) such that M ends in w, and $Q = R$ holds where M starts. First, we simplify $\langle M|(Q = R)? \ = \ \mathrm{Dom}(\mathrm{Img}((Q = R)\,;M)) \ = \ \mathrm{Img}((Q = R)\,;M) \ = \curvearrowleft\curvearrowleft((Q = R)\,;M)$. Evaluation of a rich test reduces to its evaluation over a trace: $[\![\curvearrowleft\curvearrowleft\ \alpha]\!]_T = \{(w,w) \mid (w,w) \in [\![\curvearrowleft\curvearrowleft\ \alpha]\!]_T^w\}$. From the definition of the semantics of $\curvearrowleft$, we have $[\![\curvearrowleft\curvearrowleft\ \alpha]\!]_T^{w_n} = \{(w_t,w_t) \mid \ w_t \sqsubseteq w_n \wedge \exists w' \sqsubseteq w_n \ \ (w',w_t) \in [\![\alpha]\!]_T^{w_n}\}$. To apply this expression, we identify both w_n and w_t with w, and α with $((Q = R)\,;M)$, and evaluate $(w',w) \in [\![(Q = R)\,;M]\!]_T^w$, where $w' \sqsubseteq w$. Evaluating the composition requires the existence of w'' where $(Q = R)$ ends and M begins; and after applying the semantics of Equality Check, we observe that w'' must be equal to w'. Thus, the semantics of the original expression gives us (w,w) such that M makes a transition from w' to w, where $w' \sqsubseteq w$, and at w', the equality holds.

Programming Constructs It is well-known that in Propositional Dynamic Logic [14], imperative programming constructs are definable using a fragment of regular languages, see the Dynamic Logic book by Harel, Kozen and Tiuryn [21]. The corresponding language is called *Deterministic Regular (While) Programs* in [21].[4] In our case, imperative constructs are definable by:

$$\begin{array}{ll} \mathbf{skip} \ := \ \mathrm{id}, \quad \mathbf{fail} \ := \curvearrowright \mathrm{id}, & \mathbf{while}\ \phi\ \mathbf{do}\ \alpha := (\phi?\,;\,\alpha)^{\uparrow}\,;\,(\curvearrowright \phi?), \\ \mathbf{if}\ \phi\ \mathbf{then}\ \alpha\ \mathbf{else}\ \beta \ := \ (\phi?\,;\,\alpha) \sqcup \beta, & \mathbf{repeat}\ \alpha\ \mathbf{until}\ \phi := \alpha\,;\,((\curvearrowright \phi?)\,;\,\alpha)^{\uparrow}\,;\,\phi?. \end{array}$$

Thus, importantly, the non-determinism of regular languages' operations $*$ and $\cup$ is not needed to formalize these programming constructs.

[4] Please note that Deterministic Regular expressions and the corresponding Glushkov automata are unrelated to what we study here. In those terms, expression $a\,;a^*$ is Deterministic Regular, while $a^*\,;a$ is not. Both expressions are not in our language.

7 Satisfaction Relation and Complexity of Query Evaluation

We now define binary and unary satisfaction relations and our main query. Using this query, we formalize the notion of a *computational problem specified by an algebraic term*, and its certificates. We analyze the data complexity [32] of the main query (i.e., when the query is fixed and structures vary). We prove that it is NP-time in the size of the input structure.

Definition 5. *Logic $\mathbb{L}$ is the logic (5) (equivalently, (3)) with* SM-PP *atomic transitions (see Definition 4). An $\mathbb{L}$-term is a term in this logic.*

Definition 6. *Let α be an $\mathbb{L}$-term. We say that pair of strings (w, w'),* satisfies *α under semantic context T if $(w, w') \in [\![\alpha]\!]_T$. In symbols: $(w, w') \models_T \alpha \;\Leftrightarrow\; (w, w') \in [\![\alpha]\!]_T$. For* state formulae *$\phi$ in (5), we define: $w \models_T \phi \;\Leftrightarrow\; (w, w) \in [\![\phi]\!]_T$.*

Suppose $(v, w) \models_T \alpha$, for some w, and $v = \mathfrak{A}$. Then we write our *main query*

$$\mathfrak{A} \models_T |\alpha\rangle\!\rangle \mathbf{T} \tag{6}$$

to denote that **there is a successful execution of α starting at $\mathfrak{A}$**. Moreover, the string w is a trace that starts at $\mathfrak{A}$ and **witnesses** (6). Inside the modality $|\alpha\rangle\!\rangle$, we have, essentially, an imperative program. We use the query (6) to formalize computational problems.

Definition 7. *A* computational problem $\mathcal{P}_\alpha$ specified by α *is an isomorphism-closed class of structures (set if the domain is known): $\mathcal{P}_\alpha := \{\mathfrak{A} \mid \mathfrak{A} \models_T |\alpha\rangle\!\rangle \mathbf{T}\}$.*

Definition 8. *The set of strings $H^\alpha_\mathfrak{A} := \{w \mid (\mathfrak{A}, w) \models_T \alpha\}$ is a* set of certificates *for the membership $\mathfrak{A} \in \mathcal{P}_\alpha$.*

Example 8. We continue with Example 7 adapting the informal language of Register Machines (Section 4). Suppose we have registers P, Q and R. Let, in structure $\mathfrak{A}$, P and Q contain a, and R contains b. Let M modify the contents of Q by copying R, i.e., $M(Q, \underline{R}) := \{\, Q(x) \leftarrowtail \underline{R(x)} \,\}$. Then $\mathfrak{A} \models_T |M\,; \langle M|(Q = R)?\rangle\!\rangle \mathbf{T}$, and string $\mathfrak{A} \cdot \mathfrak{B}$ witnesses it, where $\mathfrak{B}$ is the result of performing M in $\mathfrak{A}$.

Theorem 1. *The data complexity of checking $\mathfrak{A} \models_T |\alpha\rangle\!\rangle \mathbf{T}$, where the $\mathbb{L}$-term α and the semantic context T are fixed, and input structures $\mathfrak{A}$ vary, is in* NP.

Proof (Outline). We observe that, for fixed α and T, the number of states visited during a computation can be at most polynomial in the size of the input structure. This is because (1) Maximum Iterate disregards cycles (if there is a loop, there is no model), and (2) the total number of states in a transition system can be at most polynomial in the size of the input structure (all predicates that change from state to state are singleton-set). We guess a sequence of τ-structures of length n^k, where $n = dom(\mathfrak{A})$. Then we verify whether the sequence is a witness for the main query. We argue that SM-PP atomic modules with rules of the form (3) can be evaluated in P-time, and the algebraic operations preserve this property. For a fixed formula, there could be only a constant number of loops. Details will be given in the full version of the paper.

Thus, if we know what choices have been made, verifying $\mathfrak{A} \models_T |\alpha\rangle\rangle \mathbf{T}$, which is our main task, is in P-time. But, there could be exponentially many such sequences of choices in general. Thus, part (i) of Definition 2 of a logic for P-time is not yet satisfied. To complete the work, we need to find decidable conditions under which the computation can proceed by making guesses at each step *arbitrarily*, i.e., by using *any* certificate in $H^{\alpha}_{\mathfrak{A}}$. Thus, we need to study symmetries among the elements of the set $\mathcal{L}(\alpha) := \bigcup_{\mathfrak{A} \in \mathcal{P}_\alpha} H^{\alpha}_{\mathfrak{A}}$, which is a *formal language specified by* α.

8 Encoding P-time Turing Machines

In this section, we demonstrate that our logic is strong enough to encode any P-time Turing machine over unordered structures. More specifically, we show that, for every polynomial-time recognizable class $\mathcal{K}$ of structures, there is a sentence of logic $\mathbb{L}$ (introduced in Definition 5) whose models are exactly $\mathcal{K}$. This corresponds to part (ii) of Definition 2 of a logic for P-time.

We focus on the boolean query (6) from Section 7 and outline such a construction. The main idea is that a linear order on domain elements is *induced* by a path in a Kripke structure, that is, by a sequence of atomic choices. In this path, we produce new elements one by one, as in the EVEN example. The linear order corresponds to an order on the tape of a Turing machine. After such an order is guessed, a deterministic computation, following the path, proceeds for that specific order. Note that, similarly to Theorem 1, one arbitrary sequence of choices is enough to generate a specific order on a tape of a deterministic Turing machine. We now explain this construction. We assume, without loss of generality that the machine runs for n^k steps, where n is the size of the domain. The program is of the form:

$$\alpha_{\mathrm{TM}}(\mathfrak{A}) := \mathrm{ORDER} \ ; \ \mathrm{START} \ ; \ \mathbf{repeat} \ \mathrm{STEP} \ \mathbf{until} \ \mathrm{END}.$$

Procedure ORDER: Guessing an order is perhaps the most important part of our construction. We use a secondary numeric domain with a linear ordering. We guess elements one-by-one, as in the EVEN example, and associate an element of the primary domain with an element of the secondary one, using co-existence in the same structure. Each path corresponds to a possible linear ordering.[5] Once an ordering is produced, the primary domain elements are no longer needed. Later, in START procedure and in the main loop, we use k-tuples of the elements of the secondary domain for positions on the tape and time, to count the steps in the computation. We use the lexicographic ordering on these tuples. "Next in the lexicographic ordering" relation can be produced on the fly, using tuples of unary definable relations, that change from state to state.

Procedure START: This procedure creates an encoding of the input structure $\mathfrak{A}$ (say, a graph) in a sequence of structures in the transition system, to mimic an encoding $enc(\mathfrak{A})$ on a tape of a Turing machine. We use structures to represent cells of the tape of the Turing machine (one τ-structure = one cell). The procedure follows a specific path, and thus a specific order generated by the procedure START. An element is smaller in

[5] Note that the order is defined *not* with respect to an input Tarski structure, but with respect to the Kripke structure where Tarski (i.e., FO) structures are domain elements.

that order if it was generated earlier, which can be checked using Comparisons Extended in Time. Subprocedure $Encode(\underline{vocab(\mathfrak{A})}, \ldots, S_\sigma, \ldots, \bar{P}, \ldots)$ operates as follows. In every state (= cell), it keeps the input structure $\mathfrak{A}$ itself, and adds the part of the encoding of $\mathfrak{A}$ that belongs to that cell. The interpretations of $\bar{P}$ over the secondary domain of labels provide cell positions on the simulated input tape. Each particular encoding is done for a specific induced order on domain elements, in the path that is being followed.

In addition to producing an encoding, the procedure START sets the state of the Turing machine to be the initial state Q_0. It also sets initial values for the variables used to encode the instructions of the Turing machine.

Expression START is similar to the first-order formula $\beta_\sigma(\bar{a})$ used by Grädel in his proof of capturing P-time using SO-HORN logic on ordered structures [15]. The main difference is that instead of tuples of domain elements $\bar{a}$ used to refer to the addresses of the cells on a tape, we use tuples $\bar{P}$, also of length k. Grädel's formula $\beta_\sigma(\bar{a})$ for encoding input structures has the following property: $(\mathfrak{A}, <) \models \beta_\sigma(\bar{a}) \quad \Leftrightarrow$ the $\bar{a}$-th symbol of $enc(\mathfrak{A})$ is σ. Here, we have:

$$(\mathfrak{A}, w) \models_T Encode(\ldots, S_\sigma, \ldots, P_1(a_1), \ldots, P_k(a_k), \ldots)$$
$$\Leftrightarrow \quad \text{the } P_1(a_1), \ldots, P_k(a_k)\text{-th symbol of } enc(\mathfrak{A}) \text{ is } \sigma,$$

where $\bar{a}$ is a tuple of elements of the secondary domain, w is a string that starts in input structure $\mathfrak{A}$ and induces a linear order on the input domain through an order on structures (states in the transition system $\mathcal{R}_\mathfrak{A}$ rooted in $\mathfrak{A}$). That specific generated order is used in the encoding of the input structure. Another path produces a different order, and constructs an encoding for that order.

Procedure STEP: This procedure encodes the instructions of the deterministic Turing machine. SM-PP restrictions of Conjunctive Queries are well-suited for this purpose. Instead of time and tape positions as arguments of binary predicates as in Fagin's [13] and Grädel's [15] proofs, we use coexistence in the same structure with k-tuples of domain elements, as well as lexicographic successor and predecessor on such tuples.Polynomial time of the computation is guaranteed because time, in the repeat-until part, is tracked with k-tuples of domain elements.

Procedure END: It checks if the accepting state of the Turing machine is reached.

We have that, for any P-time Turing machine, we can construct term α_{TM} in the logic $\mathbb{L}$ introduced in Definition 5, such that $\mathfrak{A} \models_T |\alpha_{\mathrm{TM}}\rangle\rangle \mathbf{T}$ if and only if the Turing machine accepts an encoding of $\mathfrak{A}$ for some specific but arbitrary order of domain elements on its tape.

9 Conclusion

Motivated by the central quest for a logic for PTIME in Descriptive Complexity, we have defined a modal logic that can refer back in the executions. The logic is, simultaneously, an algebra of binary relations and a modal Dynamic Logic. While the algebraic operations are rather restrictive, the logic defines the main constructs of Imperative programming. Atomic transitions are defined to be a singleton-set restriction of Monadic Conjuctive Queries. These queries, intuitively, represent non-deterministic conditional assignments. Together with algebraic terms used as control, this gives us a machine

model. We have demonstrated that any P-time Turing machine can be simulated in the logic. Counting properties on unordered structures can be expressed, even though the logic does not have a special cardinality construct. For example, to count to four, we guess a new element exactly four times. Another example of a cardinality property is the query EVEN from Example 6. The logic also expresses reachability type of queries. For example, to encode s-t-Connectivity of a graph, starting from s, we use an atomic module to non-deterministically select what edge of the input graph to follow, and repeat the process until t is reached. While in general, model checking is in NP, the s-t-Connectivity encoding can be evaluated in P-time. Moreover, the order in which the edges of the graph are traversed does not matter.

The next step of this research is to understand *under what general syntactic conditions on the terms α of the logic $\mathbb{L}$, evaluating the main query (6), i.e.,*

$$\mathfrak{A} \models_T \ |\alpha\rangle\rangle \mathbf{T}$$

can be done naively, that is, by following one (any) sequence of atomic choices. The logic for P-time (Definition 2) would be the logic $\mathbb{L}$ introduced in Definition 5, plus these decidable conditions. Identifying these conditions is a highly non-trivial mathematical task that involves a study of an automorphism structure of the formal language $\mathcal{L}(\alpha)$ specified by α. The proposed Dynamic Logic makes algebraic tools for such a study available.

References

1. Atserias, A., Bulatov, A.A., Dawar, A.: Affine systems of equations and counting infinitary logic. Theor. Comput. Sci. **410**(18), 1666–1683 (2009)
2. Atserias, A., Dawar, A., Ochremiak, J.: On the power of symmetric linear programs. In: LICS. pp. 1–13. IEEE (2019)
3. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
4. Blass, A., Gurevich, Y., Shelah, S.: Choiceless polynomial time. Ann. Pure Appl. Logic **100**(1-3), 141–187 (1999)
5. Blass, A., Gurevich, Y., Shelah, S.: On polynomial time computation over unordered structures. Journal of Symbolic Logic **60**(3), 1093–1125 (2002)
6. Bulatov, A.A.: A dichotomy theorem for nonuniform CSPs. In: FOCS. pp. 319–330. IEEE Computer Society (2017)
7. Cai, J., Fürer, M., Immerman, N.: An optimal lower bound on the number of variables for graph identifications. Combinatorica **12**(4), 389–410 (1992)
8. Chandra, A.K., Harel, D.: Structure and complexity of relational queries. J. Comput. Syst. Sci. **25**(1), 99–128 (1982)
9. Dawar, A.: On complete problems, relativizations and logics for complexity classes. In: Fields of Logic and Computation, Essays Dedicated to Yuri Gurevich on the Occasion of His 70th Birthday. pp. 201–207 (2010)
10. Dawar, A., Grohe, M., Holm, B., Laubner, B.: Logics with rank operators. In: LICS. pp. 113–122. IEEE Computer Society (2009)
11. Dawar, A., Wilsenach, G.: Symmetric arithmetic circuits. In: ICALP. LIPIcs, vol. 168, pp. 36:1–36:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)

12. Enderton, H.B.: A mathematical introduction to logic. Academic Press (1972)
13. Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. Complexity of computation, SIAM-AMC proceedings **7**, 43–73 (1974)
14. Fischer, M.J., Ladner, R.E.: Propositional dynamic logic of regular programs. J. Comput. Syst. Sci. **18**(2), 194–211 (1979)
15. Grädel, E.: Capturing complexity classes by fragments of second order logic. In: Computational Complexity Conference. pp. 341–352. IEEE Computer Society (1991)
16. Grädel, E.: Why are modal logics so robustly decidable? In: Current Trends in Theoretical Computer Science, pp. 393–408 (2001)
17. Grädel, E., Pakusa, W.: Rank logic is dead, long live rank logic! J. Symb. Log. **84**(1), 54–87 (2019)
18. Grädel, E., Pakusa, W., Schalthöfer, S., Kaiser, L.: Characterising choiceless polynomial time with first-order interpretations. In: Proc. 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '15) (2015)
19. Grohe, M.: The quest for a logic capturing PTIME. In: LICS. pp. 267–271. IEEE Computer Society (2008)
20. Gurevich, Y.: Logic and the challenge of computer science. In: Börger, E. (ed.) Current Trends in Theoretical Computer Science, pp. 1–57. Computer Science Press (1988)
21. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic (Foundations of Computing). MIT Press (2000)
22. Immerman, N.: Relational queries computable in polynomial time. Inf. Control. **68**(1-3), 86–104 (1986)
23. Immerman, N.: Descriptive complexity. Graduate texts in computer science, Springer (1999)
24. Jónsson, B., Tarski, A.: Representation problems for relation algebras. Bull. Amer. Math. Soc. **74**, 127–162 (1952)
25. Lichter, M.: Separating rank logic from polynomial time. In: LICS. pp. 1–13. IEEE (2021)
26. McCarthy, J., Hayes, P.J.: Some philosophical problems from the standpoint of artificial intelligence. In: Meltzer, B., Michie, D. (eds.) Machine Intelligence 4, pp. 463–502. Edinburgh University Press (1969)
27. Pakusa, W., Schalthöfer, S., Selman, E.: Definability of cai-fürer-immerman problems in choiceless polynomial time. ACM Trans. Comput. Log. **19**(2), 7:1–7:27 (2018)
28. Segoufin, L., ten Cate, B.: Unary negation. Log. Methods Comput. Sci. **9**(3) (2013)
29. Shepherdson, J.C., Sturgis, H.E.: Computability of recursive functions. J. ACM **10**(2), 217–255 (1963)
30. Ternovska, E.: An algebra of modular systems: Static and dynamic perspectives. In: Proceedings of the 12th International Symposium on Frontiers of Combining Systems (FroCoS) (September 2019)
31. Vardi, M.: On the complexity of bounded-variable queries. In: PODS. pp. 266–276. ACM Press (1995)
32. Vardi, M.Y.: The complexity of relational query languages (extended abstract). In: STOC. pp. 137–146. ACM (1982)
33. Vardi, M.Y.: Why is modal logic so robustly decidable? In: Descriptive Complexity and Finite Models, Proceedings of a DIMACS Workshop 1996, Princeton, New Jersey, USA, January 14-17, 1996. pp. 149–184 (1996)
34. Zeume, T., Harwath, F.: Order-invariance of two-variable logic is decidable. In: LICS. pp. 807–816. ACM (2016)
35. Zhuk, D.: A proof of CSP dichotomy conjecture. In: FOCS. pp. 331–342. IEEE Computer Society (2017)

A Labelled Sequent Calculus for Public Announcement Logic

Hao Wu[1], Hans van Ditmarsch[2], and Jinsheng Chen[3]

[1] Institute of Logic and Cognition, Sun Yat-sen University
[2] Department of Computer Science, Open University of the Netherlands
[3] School of Philosophy, Zhejiang University

Abstract. We propose a cut-free labelled sequent calculus for PAL, which is an extension of that for EL with sequent rules adapted from the reduction axioms. This calculus admits cut and allows terminating proof search.

Keywords: PAL, labelled sequent calculus, cut elimination.

1 Introduction

The Hilbert-style axiomatization of PAL is obtained by adding to that of EL the so called *reduction axioms* for announcement operators, which can be used to eliminate announcement operators in a PAL-formula. Generally speaking, it is difficult to prove whether proof search using a Hilbert-style axiomatization is decidable. In view of these, many proof systems for PAL are proposed in the literature, e.g., tableau systems [2], labelled sequent calculi [1, 6]. Here we propose another labelled sequent calculus for PAL, which is based on a labelled sequent calculus for EL proposed in Hakli and Negri [3] and rules for announcement operators designed according to the reduction axioms. This calculus admits structural rules (including cut) and allows terminating proof search. Moreover, the calculus is based on the original semantics and takes into account the conditions of reflexivity, transitivity and symmetry in EL.

2 Labelled sequent calculus for EL and PAL

2.1 EL and PAL

Given a denumerable set Prop of variables and a finite set Ag of agents. Language $\mathcal{L}_{\mathsf{EL}}$ for epistemic logic is defined inductively as follows:

$$\mathcal{L} ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \to \varphi \mid K_a\varphi$$

Language $\mathcal{L}_{\mathsf{PAL}}$ for public announcement logic is $\mathcal{L}_{\mathsf{EL}}$ plus the public announcement formula $[\varphi]\varphi$, where $p \in \mathsf{Prop}$ and $a \in \mathsf{Ag}$. Important definitions of *epistemic frame*, *epistemic model*, *restricted model*, *model validity* and *satisfiability*

can be found in [7]. EL is axiomatized by (Tau), (K), (4), (T), (5), (MP) and (GK_a). PAL is axiomatized by the axiomatization for EL plus *reduction axioms* (R1–6)[7].

A labelled sequent calculus for a logic with Kripke semantics is based on the internalization of Kripke semantics. Given an epistemic frame $\mathcal{F} = (W, \{\sim_a\}_{a\in\mathsf{Ag}})$, a *relational atom* is of the form $x \sim_a y$, where $x, y \in W$ and $a \in \mathsf{Ag}$. A *labelled formula* is of the form $x : \varphi$, where $x \in W$ and φ is an $\mathcal{L}_{\mathsf{EL}}$-formula.

Given an epistemic model $\mathcal{M} = (W, \{\sim_a\}_{a\in\mathsf{Ag}}, V)$, the *interpretation* $\tau_{\mathcal{M}}$ of relational atoms and labelled formulas are defined as follows:

$$\begin{array}{rcl} \tau_{\mathcal{M}}(x \sim_a y) & = & x \sim_a y; \\ \tau_{\mathcal{M}}(x : \varphi) & = & \mathcal{M}, x \Vdash \varphi. \end{array}$$

Validity of a labelled sequent and a sequent rule $\mathcal{R}$ is conventionally defined.

2.2 Labelled sequent calculus G_{EL} for EL

Definition 1 *G_{EL} consists of the following initial sequents and rules:*

(1) Initial sequents:

$$x : p, \Gamma \Rightarrow \Delta, x : p \qquad\qquad x \sim_a y, \Gamma \Rightarrow \Delta, x \sim_a y$$

(2) Propositional rules:

$$(\neg\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, x : \varphi}{x : \neg\varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\neg)\ \frac{x : \varphi, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, x : \neg\varphi}$$

$$(\wedge\Rightarrow)\ \frac{x : \varphi_1, x : \varphi_2, \Gamma \Rightarrow \Delta}{x : \varphi_1 \wedge \varphi_2, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\wedge)\ \frac{\Gamma \Rightarrow \Delta, x : \varphi_1 \quad \Gamma \Rightarrow \Delta, x : \varphi_2}{\Gamma \Rightarrow \Delta, x : \varphi_1 \wedge \varphi_2}$$

$$(\rightarrow\Rightarrow)\ \frac{\Gamma \Rightarrow \Delta, x : \varphi \quad x : \psi, \Gamma \Rightarrow \Delta}{x : \varphi \rightarrow \psi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow\rightarrow)\ \frac{x : \varphi, \Gamma \Rightarrow \Delta, x : \psi}{\Gamma \Rightarrow \Delta, x : \varphi \rightarrow \psi}$$

(3) Modal rules:

$$(K_a \Rightarrow)\ \frac{y : \varphi, x : K_a\varphi, x \sim_a y, \Gamma \Rightarrow \Delta}{x : K_a\varphi, x \sim_a y, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow K_a)\ \frac{x \sim_a y, \Gamma \Rightarrow \Delta, y : \varphi}{\Gamma \Rightarrow \Delta, x : K_a\varphi}$$

where y does not occur in the conclusion of $(\Rightarrow K_a)$, .

(4) Relational rules:

$$(Ref_a)\ \frac{x \sim_a x, \Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta} \qquad (Trans_a)\ \frac{x \sim_a z, x \sim_a y, y \sim_a z, \Gamma \Rightarrow \Delta}{x \sim_a y, y \sim_a z, \Gamma \Rightarrow \Delta}$$

$$(Sym_a)\ \frac{y \sim_a x, x \sim_a y, \Gamma \Rightarrow \Delta}{x \sim_a y, \Gamma \Rightarrow \Delta}$$

Proposition 2 *For any $\mathcal{L}_{\mathsf{EL}}$-formula φ, $G_{\mathsf{EL}} \vdash x : \varphi, \Gamma \Rightarrow \Delta, x : \varphi$.*

By proofs similar to those in [5], we have Theorems 3 and 4.

Theorem 3 *Structural rules* $(w\Rightarrow)$, $(\Rightarrow w)$, $(c\Rightarrow)$, $(\Rightarrow c)$, $(c_R\Rightarrow)$ *and* $(\Rightarrow c_R)$ *are height-preserving admissible in* G_{EL}. *The cut rule* (Cut) *is admissible in* G_{EL}.

$$(w\Rightarrow)\ \frac{\Gamma\Rightarrow\Delta}{x:\varphi,\Gamma\Rightarrow\Delta} \qquad (\Rightarrow w)\ \frac{\Gamma\Rightarrow\Delta}{\Gamma\Rightarrow\Delta,x:\varphi}$$

$$(c\Rightarrow)\ \frac{x:\varphi,x:\varphi,\Gamma\Rightarrow\Delta}{x:\varphi,\Gamma\Rightarrow\Delta} \qquad (\Rightarrow c)\ \frac{\Gamma\Rightarrow\Delta,x:\varphi,x:\varphi}{\Gamma\Rightarrow\Delta,x:\varphi}$$

$$(c_R\Rightarrow)\ \frac{x\sim_a y,x\sim_a y,\Gamma\Rightarrow\Delta}{x\sim_a y,\Gamma\Rightarrow\Delta} \qquad (\Rightarrow c_R)\ \frac{\Gamma\Rightarrow\Delta,x\sim_a y,x\sim_a y}{\Gamma\Rightarrow\Delta,x\sim_a y}$$

$$(Cut)\ \frac{\Gamma\Rightarrow\Delta,x:\varphi \quad x:\varphi,\Gamma'\Rightarrow\Delta'}{\Gamma,\Gamma'\Rightarrow\Delta,\Delta'}$$

Theorem 4 G_{EL} *allows terminating proof search.*

2.3 Labelled sequent calculus G_{PAL} for PAL

Definition 5 G_{PAL} *is* G_{EL} *plus the following sequent rules:*

$$(R1\Rightarrow)\ \frac{\Gamma\Rightarrow\Delta,x:\varphi \quad x:p,\Gamma\Rightarrow\Delta}{x:[\varphi]p,\Gamma\Rightarrow\Delta} \qquad (\Rightarrow R1)\ \frac{x:\varphi,\Gamma\Rightarrow\Delta,x:p}{\Gamma\Rightarrow\Delta,x:[\varphi]p}$$

$$(R2\Rightarrow)\ \frac{\Gamma\Rightarrow\Delta,x:\varphi \quad x:\neg[\varphi]\psi,\Gamma\Rightarrow\Delta}{x:[\varphi]\neg\psi,\Gamma\Rightarrow\Delta} \qquad (\Rightarrow R2)\ \frac{x:\varphi,\Gamma\Rightarrow\Delta,x:\neg[\varphi]\psi}{\Gamma\Rightarrow\Delta,x:[\varphi]\neg\psi}$$

$$(R3\Rightarrow)\ \frac{x:[\varphi]\psi_1,x:[\varphi]\psi_2,\Gamma\Rightarrow\Delta}{x:[\varphi](\psi_1\wedge\psi_2),\Gamma\Rightarrow\Delta} \qquad (\Rightarrow R3)\ \frac{\Gamma\Rightarrow\Delta,x:[\varphi]\psi_1 \quad \Gamma\Rightarrow\Delta,x:[\varphi]\psi_2}{\Gamma\Rightarrow\Delta,x:[\varphi](\psi_1\wedge\psi_2)}$$

$$(R4\Rightarrow)\ \frac{\Gamma\Rightarrow\Delta,x:[\varphi]\psi_1 \quad x:[\varphi]\psi_2,\Gamma\Rightarrow\Delta}{x:[\varphi](\psi_1\to\psi_2),\Gamma\Rightarrow\Delta} \qquad (\Rightarrow R4)\ \frac{x:[\varphi]\psi_1,\Gamma\Rightarrow\Delta,x:[\varphi]\psi_2}{\Gamma\Rightarrow\Delta,[\varphi](\psi_1\to\psi_2)}$$

$$(R5\Rightarrow)\ \frac{\Gamma\Rightarrow\Delta,x:\varphi \quad K_a[\varphi]\psi,\Gamma\Rightarrow\Delta}{x:[\varphi]K_a\psi,\Gamma\Rightarrow\Delta} \qquad (\Rightarrow R5)\ \frac{x:\varphi,\Gamma\Rightarrow\Delta,x:K_a[\varphi]\psi}{\Gamma\Rightarrow\Delta,x:[\varphi]K_a\psi}$$

$$(R6\Rightarrow)\ \frac{x:[\varphi\wedge[\varphi]\psi]\chi,\Gamma\Rightarrow\Delta}{x:[\varphi][\psi]\chi,\Gamma\Rightarrow\Delta} \qquad (\Rightarrow R6)\frac{\Gamma\Rightarrow\Delta,x:[\varphi\wedge[\varphi]\psi]\chi}{\Gamma\Rightarrow\Delta,x:[\varphi][\psi]\chi}$$

Another desirable property for sequent rules is that the complexity of each premise should be less than that of the conclusion, which can be guaranteed by the following definition:

Definition 6 *Let* φ *be an* $\mathcal{L}_{\mathsf{PAL}}$ *formula, The* complexity $c(\varphi)$ *of* φ *is defined as follows:*

$$c(p)=1 \qquad c(\varphi\to\psi)=1+\max\{c(\varphi),c(\psi)\}$$
$$c(\neg\varphi)=1+c(\varphi) \qquad c(K_a\varphi)=1+c(\varphi)$$
$$c(\varphi\wedge\psi)=1+\max\{c(\varphi),c(\psi)\} \qquad c([\varphi]\psi)=(4+c(\varphi))\cdot c(\psi).$$

Then we have the following lemma:

Lemma 7 *For all* $\mathcal{L}_{\mathsf{PAL}}$*-formulas* φ,ψ *and* χ*:*

(1) $c([\varphi]p)>c(\varphi\to p)$;
(2) $c([\varphi]\neg\psi)>c(\varphi\to\neg[\varphi]\psi)$;
(3) $c([\varphi](\psi\wedge\chi))>c([\varphi]\psi\wedge[\varphi]\chi)$;
(4) $c([\varphi]K_a\psi)>c(\varphi\to K_a[\varphi]\psi)$;
(5) $c([\varphi][\psi]\chi)>c([\varphi\wedge[\varphi]\psi]\chi)$.

Lemma 8 *For any* $\mathcal{L}_{\mathsf{PAL}}$*-formula* φ, $G_{\mathsf{PAL}}\vdash x:\varphi,\Gamma\Rightarrow\Delta,x:\varphi$.

3 Admissibility of some structural rules

In light of the reduction axioms, we can define a translation from $\mathcal{L}_{\mathsf{PAL}}$-formulas to $\mathcal{L}_{\mathsf{EL}}$-formulas

Definition 9 *The translation $t : \mathcal{L}_{\mathsf{PAL}} \to \mathcal{L}_{\mathsf{EL}}$ is defined as follows:*

$$\begin{array}{lcl} t(p) & = & p \\ t(\neg\varphi) & = & \neg t(\varphi) \\ t(\varphi \wedge \psi) & = & t(\varphi) \wedge t(\psi) \\ t(\varphi \to \psi) & = & t(\varphi) \to t(\psi) \\ t(K_a\varphi) & = & K_a t(\varphi) \end{array} \qquad \begin{array}{lcl} t([\varphi]p) & = & t(\varphi \to p) \\ t([\varphi]\neg\psi) & = & t(\varphi \to \neg[\varphi]\psi) \\ t([\varphi](\psi \wedge \chi)) & = & t([\varphi]\psi \wedge [\varphi]\chi) \\ t([\varphi](\psi \to \chi)) & = & t([\varphi]\psi \to [\varphi]\chi) \\ t([\varphi]K_a\psi) & = & t(\varphi \to K_a[\varphi]\psi) \\ t([\varphi][\psi]\chi) & = & t([\varphi \wedge [\varphi]\psi]\chi) \end{array}$$

Now we extend translation t to relational atoms and labelled $\mathcal{L}_{\mathsf{PAL}}$-formulas: for any relational atom $x \sim_a y$, let $t(x \sim_a y) = x \sim_a y$; for any labelled $\mathcal{L}_{\mathsf{PAL}}$-formula $x : \varphi$, $t(x : \varphi) = x : t(\varphi)$. Moreover, for any set Γ of relational atoms and labelled formulas: $t(\Gamma) = \{t(\sigma) \mid \sigma \in \Gamma\}$.

Lemma 10 *For any $\mathcal{L}_{\mathsf{PAL}}$-sequent $x : \varphi, \Gamma \Rightarrow \Delta$, the following hold:*

(1) if $G_{\mathsf{PAL}} \vdash x : t(\varphi), t(\Gamma) \Rightarrow t(\Delta)$, then $G_{\mathsf{PAL}} \vdash x : \varphi, t(\Gamma) \Rightarrow t(\Delta)$;
(2) if $G_{\mathsf{PAL}} \vdash t(\Gamma) \Rightarrow t(\Delta), x : t(\varphi)$, then $G_{\mathsf{PAL}} \vdash t(\Gamma) \Rightarrow t(\Delta), x : \varphi$.

The following theorem is a bridge between G_{PAL} and G_{EL}, enabling us to prove properties of G_{PAL} through G_{EL}.

Theorem 11 *For any $\mathcal{L}_{\mathsf{PAL}}$-sequent $\Gamma \Rightarrow \Delta$,*

(1) if $G_{\mathsf{EL}} \vdash t(\Gamma) \Rightarrow t(\Delta)$, then $G_{\mathsf{PAL}} \vdash \Gamma \Rightarrow \Delta$;
(2) if $G_{\mathsf{PAL}} \vdash_h \Gamma \Rightarrow \Delta$, then $G_{\mathsf{EL}} \vdash_h t(\Gamma) \Rightarrow t(\Delta)$.

Corollary 12 *The following structural rules are admissible in G_{PAL}:*

$$(w \Rightarrow)\ \frac{\Gamma \Rightarrow \Delta}{x : \varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow w)\ \frac{\Gamma \Rightarrow \Delta}{\Gamma \Rightarrow \Delta, x : \varphi}$$

$$(c \Rightarrow)\ \frac{x : \varphi, x : \varphi, \Gamma \Rightarrow \Delta}{x : \varphi, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow c)\ \frac{\Gamma \Rightarrow \Delta, x : \varphi, x : \varphi}{\Gamma \Rightarrow \Delta, x : \varphi}$$

$$(c_R \Rightarrow)\ \frac{x \sim_a y, x \sim_a y, \Gamma \Rightarrow \Delta}{x \sim_a y, \Gamma \Rightarrow \Delta} \qquad (\Rightarrow c_R)\ \frac{\Gamma \Rightarrow \Delta, x \sim_a y, x \sim_a y}{\Gamma \Rightarrow \Delta, x \sim_a y}$$

$$(Cut)\ \frac{\Gamma \Rightarrow \Delta, x : \varphi \quad x : \varphi, \Gamma' \Rightarrow \Delta'}{\Gamma, \Gamma' \Rightarrow \Delta, \Delta'}$$

Theorem 13 (Soundness and Completeness) *For any $\mathcal{L}_{\mathsf{PAL}}$-formula φ, $\varphi \in \mathsf{PAL}$ iff $G_{\mathsf{PAL}} \vdash \Rightarrow \varphi$.*

Decidability can be proved conventionally by Theorems 4 and 11 in an indirect way. A direct proof is possible by extending the notion of *subformulas* and *minimal derivations* to exclude the four potential sources of non-terminating proof search.

4 Comparison

Existing works on labelled sequent calculi for PAL are based on a reformulation of the semantics for PAL. The notion of *model restriction* can be generalized to a list of formulas. Denote by α / β finite lists $(\varphi_1, \ldots, \varphi_n)$ of formulas, and by ϵ the empty list. For any list $\alpha = (\varphi_1, \ldots, \varphi_n)$ of formulas, $\mathcal{M}^\alpha$ is defined recursively as follows: $\mathcal{M}^\alpha := \mathcal{M}$ (if $\alpha = \epsilon$), and $\mathcal{M}^\alpha := (\mathcal{M}^\beta)^{\varphi_n} = (W^{\beta,\varphi_n}, (\sim_a^{\beta,\varphi_n})_{a\in\mathsf{Ag}}, V^{\beta,\varphi_n})$ (if $\alpha = \beta, \varphi_n$).

Then an equivalent semantics for PAL is defined as follows:

$$
\begin{array}{ll}
\mathcal{M}^{\alpha,\varphi}, w \Vdash p & \text{iff } \mathcal{M}^\alpha, w \Vdash \varphi \text{ and } \mathcal{M}^\alpha, w \Vdash p \\
\mathcal{M}^\alpha, w \Vdash \neg\varphi & \text{iff } \mathcal{M}^\alpha, w \nVdash \varphi \\
\mathcal{M}^\alpha, w \Vdash \varphi \wedge \psi & \text{iff } \mathcal{M}^\alpha, w \Vdash \varphi \text{ and } \mathcal{M}^\alpha, w \Vdash \psi \\
\mathcal{M}^\alpha, w \Vdash \varphi \to \psi & \text{iff } \mathcal{M}^\alpha, w \nVdash \varphi \text{ or } \mathcal{M}^\alpha, w \Vdash \psi \\
\mathcal{M}^\alpha, w \Vdash K_a\varphi & \text{iff for all } v \in W, w \sim_a^\alpha v \text{ implies } \mathcal{M}^\alpha, v \Vdash \varphi \\
\mathcal{M}^\alpha, w \Vdash [\varphi]\psi & \text{iff } \mathcal{M}^\alpha, w \Vdash \varphi \text{ implies } \mathcal{M}^{\alpha,\varphi}, w \Vdash \psi
\end{array}
$$

With this semantics, Balbiani [1] and Nomura et al. [6]developed different labelled sequent calculi for PAL, fixing the defects in Maffezioli and Negri [4]. The calculus in Balbiani [1] admits cut and allows terminating proof search, while the calculus in Nomura et al. [6] admits cut. However, unlike ours, their calculi do not include the inference rules for S5 (i.e., (Ref_a), $(Trans_a)$ and (Sym_a)). As a result, their calculi are for PAL which is based on the smallest normal modal logic K. Moreover, our calculus is based on the original semantics for PAL and uses reduction rules transformed from reduction axioms to deal with the announcement operators.

References

1. Balbiani, P., Demange, V., Galmiche, D.: A sequent calculus with labels for pal. Advances in Modal Logic **6** (2014)
2. Balbiani, P., Van Ditmarsch, H., Herzig, A., De Lima, T.: Tableaux for public announcement logic. Journal of Logic and Computation **20**(1), 55–76 (2010)
3. Hakli, R., Negri, S.: Proof theory for distributed knowledge. In: International Workshop on Computational Logic in Multi-Agent Systems. pp. 100–116. Springer (2007)
4. Maffezioli, P., Negri, S.: A gentzen-style analysis of public announcement logic. In: Proc. of the 2nd LogKCA. pp. 293–313. University of the Basque Country Press Donostia, Spain (2010)
5. Negri, S.: Proof analysis in modal logic. Journal of Philosophical Logic **34**(5), 507–544 (2005)
6. Nomura, S., Sano, K., Tojo, S.: Revising a labelled sequent calculus for public announcement logic. In: Structural Analysis of Non-Classical Logics, pp. 131–157. Springer (2016)
7. Van Ditmarsch, H., van Der Hoek, W., Kooi, B.: Dynamic epistemic logic, vol. 337. Springer Science & Business Media (2007)