From CTL to Datalog

Foto Afrati
Dept. of Computer Science
N.T.U.A. 157 80 Zographou,
Athens, Greece
afrati@softlab.ntua.gr

Theodore Andronikos
Dept. of Computer Science
N.T.U.A. 157 80 Zographou,
Athens, Greece
thandron@softlab.ntua.gr

Vassia Pavlaki
Dept. of Computer Science
N.T.U.A. 157 80 Zographou,
Athens, Greece
vpaulaki@softlab.ntua.gr

Eugenie Foustoucos
Dept. of Computer Science
N.T.U.A.
157 80 Zographou, Athens,
Greece
eugenie@ermis.cs.ntua.gr

Irène Guessarian
Laboratory for Computer
Science,LIAFA
Université Paris 6, 4 Place
Jussieu, 75252 Paris Cedex 5
ig@liafa.jussieu.fr

ABSTRACT

We provide a translation from CTL to Datalog$^{\neg}_{Succ}$. The translation has the following advantages: a) It is natural. b) It provides intuition to the expressive power of CTL and its various fragments. c) It uses a fragment of Datalog$^{\neg}_{Succ}$ which is close to the expressive power of CTL.

Categories and Subject Descriptors
F.4 [Mathematical Logic and Formal Languages]: Mathematical Logic—temporal logic;
I.2.3 [Deduction and Theorem Proving]—logic programming

General Terms
Theory, verification

Keywords: Temporal logic, Datalog, Verification

1. INTRODUCTION

This paper relates to Paris Kanellakis research on systems verification and specification [14], [13], [12]. In [14], Paris Kanellakis and his student Scott Smolka consider Milner's Calculus for Communicating Systems (CCS) and stress its relation to the classical theory of regular expressions and finite automata. Fundamental complexity questions about CCS are answered in an elegant way. Basic notions of equivalence in CCS are treated as refinements of the classical notion of nondeterministic finite automata equivalence. Research on model checking in the modal μ-calculus is pursued by Scott Smolka in a paper in LICS94 [25] which Paris was aware of and liked. In this last paper, the connection between modal μ-calculus and Datalog$^{\neg}$ is also observed. Results about the parallel complexity of Datalog$^{\neg}$ queries and a reduction of $\wedge$-free formulas of the alternation-free modal μ-calculus to Datalog$^{\neg}$ is used to derive results about the parallel computational complexity of this fragment of modal μ-calculus. Interestingly, in the present paper, we present a reduction of CTL (Computational Tree Logic) formulas to Datalog$^{\neg}$ which is based on similar intuition as in [25]. CTL is a branching time temporal logic which is used for specification and verification of temporal properties of events [7]. We show here that the model checking problem of CTL is reduced to the query evaluation problem of Datalog$^{\neg}_{Succ}$.

Temporal logic in its most basic form corresponds to a type of logic originally developed by philosophers ([22]). Pnueli [19] was the first to propose the use of (linear) temporal logic as a language for reasoning about change over time and describing nonterminating behavior. One of the most significant developments in this area is the existence of algorithmic methods for verifying temporal-logic properties of finite state programs. The practical significance is derived mainly from the fact that many synchronization and communication protocols can be abstracted as finite-state programs. Finite-state programs can be modelled by transition systems where each state can be characterized by a fixed number of boolean atomic propositions. These systems are known as finite propositional Kripke structures. The question of checking whether the program has specific properties is reduced to that of checking whether a temporal logic formula holds on the Kripke structure that represents the program (a.k.a. *model checking*). In defining temporal logics, there are two possible views regarding the underlying nature of time. One is that of linear time: at each moment there is only one possible future (LTL Linear temporal Logic). The other is that of branching time (treelike nature): at each moment time may follow different paths which represent different possible futures ([8, 17]). The most prominent examples of this kind are CTL (Computational Tree Logic), CTL* (Full Branching Time Logic), and μ-calculus.

The translation of temporal logics to logic programming has been investigated and related work concerning translation of CTL or μ-calculus into logic programming can be found in [21, 5, 20]. In [5], the LMC project is pre-

PCK50, June 8, 2003, San Diego, California, USA.

sented which uses XSB, a tabled logic programming system that extends Prolog-style SLD resolution with tabled resolution. Model checking has been proved to be closely related to database query evaluation. The idea is based on the principle that Kripke structures can be viewed as relational databases ([11]). In that perspective, the database query language Datalog has inspired work in [10], where the language Datalog LITE is introduced, obtained by extending stratified Datalog with generalized literals. Datalog LITE is shown to subsume temporal languages such as CTL and the alternation-free μ-calculus. In fact, it is shown to have the same expressive power as the alternation-free portion of guarded fixed point logic.

The main result of the present work is a translation of CTL to $\text{Datalog}^{\neg}_{Succ}$ i.e., Datalog with negation only on the database predicates (EDB predicates) and with the successor built-in predicate. The advantage of our translation to prior related work is that a) the translation is natural b) the fragment of $\text{Datalog}^{\neg}_{Succ}$ that we use for the translation has expressive power close to the expressive power of CTL. c) As a consequence, it provides intuition into the expressiveness of CTL and its fragments. Some of the ideas used in this paper have been previously presented in our work [1].

The following two motivating examples show that, in some cases, it is easy and straightforward to translate a CTL formula into Datalog.

EXAMPLE 1.1. *This is the first motivating example for our translation techniques. Consider the CTL formula: $\varphi \equiv \mathbf{E}\bigcirc p$. It says that, considering (1) a Kripke structure $\mathcal{K}$, which is given by a set of states, the transition relation R on the states, and atomic propositions assigned to the states, and (2) an initial state s_0 of $\mathcal{K}$, then, there exists a path starting from s_0 such that the next state (after s_0) on this path is assigned the atomic proposition p. We may view the Kripke structure as a database D with unary EDB predicates for the propositions (here EDB P is associated to proposition p) and a binary EDB predicate R for the transition relation. Now the following Datalog program says that if s_0 is computed in the answers of the query $G_{\mathbf{E}(\bigcirc p)}$, then there exists a path in D starting from s_0 which in one transition step reaches a state where P is true.*

- $G_p(x) \longleftarrow P(x)$
- $G_{\mathbf{E}\bigcirc p}(x) \longleftarrow R(x,y), G_p(y)$

Whereas this is not a recursive program, when the formula contains the "until" modality, recursion is needed as is the case in the example that follows.

EXAMPLE 1.2. *Consider now the somewhat more complex formula $\varphi \equiv \mathbf{E}\bigcirc p \wedge \mathbf{E}(q\mathbf{U}t)$. The Datalog query that expresses this formula is the following.*

- $G_p(x) \longleftarrow P(x)$
- $G_{\mathbf{E}\bigcirc p}(x) \longleftarrow R(x,y), G_p(y)$
- $G_q(x) \longleftarrow Q(x)$
- $G_t(x) \longleftarrow T(x)$
- $G_{\mathbf{E}(q\mathbf{U}t)}(x) \longleftarrow T(x)$
- $G_{\mathbf{E}(q\mathbf{U}t)}(x) \longleftarrow G_q(x), R(x,y), G_{\mathbf{E}(q\mathbf{U}t)}(y)$
- $G_{\mathbf{E}\bigcirc p \wedge \mathbf{E}(q\mathbf{U}t)}(x) \longleftarrow G_{\mathbf{E}\bigcirc p}(x), G_{\mathbf{E}(q\mathbf{U}t)}(x)$

This Datalog query expresses what the CTL formula says, i.e., that from s_0 there exists a path that is assigned p on its next state and there is also a path (different or the same) such that it is assigned q along all its states up until it gets to a state that is assigned t. The first two rules express the CTL formula $\mathbf{E}\bigcirc p$ the next three rules express the CTL formula $\mathbf{E}(q\mathbf{U}t)$ and the last rule together with the other five express the CTL formula φ.

Notice that we use heavy subscripts in the predicates to facilitate the easy connection between the program predicate and the CTL formula this predicate expresses.

For our translation, the motivating observations are:

- The result in [24], [16] which proves that any CTL formula can be equivalently written in a *positive normal form* such as all negations are pushed inside in the atomic propositional symbols. The modalities that are used to facilitate the rewriting in positive normal form are $\mathbf{A}$, $\mathbf{E}$, $\mathbf{U}$ and $\widetilde{\mathbf{U}}$ (for detailed formal definitions see the Definitions Section). In this paper we refer only to CTL formulas in positive normal form.
- When a CTL formula φ in positive normal form does not contain the universal quantifier, then a simple fragment of $\text{Datalog}^{\neg}$ is sufficient to express φ. The two examples above almost illustrate all the intuition except a complication needed to express $\widetilde{\mathbf{U}}$ (for an example containing $\widetilde{\mathbf{U}}$, see Section 7).

However, $\text{Datalog}^{\neg}$ is not powerful enough to express CTL because it cannot express the universal quantifier. Hence, we need to extend it in order to capture the universal modality. The extension we propose is the introduction of a variable that ranges over another domain which is totally ordered (e.g., the integers). This variable will serve the purpose of counting the states in the Kripke structure, so that it ensures that all states reachable from a certain state are considered during an evaluation of the query on the Kripke structure. This extension can be viewed as allowing the built-in predicate *Succ* in $\text{Datalog}^{\neg}$, thus obtaining $\text{Datalog}^{\neg}_{Succ}$.

In this paragraph we explain what we mean by translation. Suppose we are given a CTL formula φ and a Kripke structure $\mathcal{K}$ and we want to check whether φ holds in $\mathcal{K}$. Then we reduce this problem to the problem: Given a $\text{Datalog}^{\neg}_{Succ}$ query and relations on the EDB predicates (structure D) find whether a certain element is an answer to the query in D. That is, we transform K into D with a special element s_0 (the root of the structure) and φ into a program Π such that φ holds in $\mathcal{K}$ iff s_0 is an answer to the query of Π on D.

Using a variant of Datalog to express temporal logics is beneficial for the following reasons:

1. $\text{Datalog}^{\neg}$ is a declarative query language with clear semantics.
2. It has polynomial data complexity, i.e., fixed $\text{Datalog}^{\neg}$ programs can be evaluated in polynomial time over the input databases.
3. $\text{Datalog}^{\neg}$ is suited for main-memory databases as well as for databases stored on disk. Sophisticated query-optimization techniques (magic-sets for instance) [2] for secondary storage access have been developed and implemented [3].

A corollary of our main result is that any CTL formula in normal form that does not use the universal quantifier can be translated in Datalog$^\neg$ such as all IDB predicates are either unary or binary, which however essentially express transitive closure. Another corollary is that any CTL formula in normal form that does not use either the universal operator or the $\widetilde{\mathbf{U}}$ modality can be translated in monadic Datalog$^\neg$, i.e., it uses only unary predicates in the head of all rules. These corollaries also help to get intuition into the expressiveness of CTL as delineate the expressiveness of its various fragments.

The rest of the paper is organized as follows. Section 2 contains the formal definition of CTL. Sections 3 contains the formal definition of Datalog$^\neg_{Succ}$. Section 4 defines temporal Kripke structures and explains how they can be viewed as relational databases. Section 5 presents the formalism of our translation and discusses the notion of equivalence between state and path CTL formulas and Datalog queries. Section 6 gives the translation rules and then proves the main result of this paper (Theorem 6.1). Section 7 gives more examples to help the reader see the intuition of the more complicated parts of the translation and Section 8 discusses possible future research directions.

2. BASIC DEFINITIONS

2.1 Syntax

In CTL we encounter two kinds of formulas: **state** and **path** formulas. CTL (Computational Tree Logic) express properties along a tree-like flow of time.

We use AP to denote the countable (and sometimes finite) set of atomic propositions: $p_0, \ldots, p_i, \ldots$.

DEFINITION 2.1. *The set of CTL **state** formulas, denoted simply CTL, is the smallest set of formulas such that:*

- *Atomic propositions, negations of atomic propositions, $\top$ and $\bot$ are CTL state formulas,*
- *If φ and ψ are CTL state formulas, then $(\varphi \vee \psi)$ and $(\varphi \wedge \psi)$ are CTL state formulas, and*
- *If φ is a* **path** *formula, then $\mathbf{E}\varphi$ and $\mathbf{A}\varphi$ are CTL state formulas.*

DEFINITION 2.2. *The set of CTL **path** formulas is the smallest set of formulas such that if φ and ψ are state formulas, then $\bigcirc\varphi$, $(\varphi \mathbf{U} \psi)$ and $(\varphi \widetilde{\mathbf{U}} \psi)$ are CTL path formulas.*

From now on, when we refer to CTL formulas without further clarification, we mean **state** formulas.

As usual one can define the following abbreviations:

- $\Diamond\varphi \equiv \top \mathbf{U} \varphi$ ("eventually"),
- $\Box\varphi \equiv \neg\Diamond\neg\varphi$ ("always"),

It should be noted here that many authors use another range of symbols for these modalities: **X** instead of $\bigcirc$, **F** instead of $\Diamond$, and **G** instead of $\Box$.

A CTL formula in which negations are applied only to atomic propositions is in **positive normal form**. In the next Section we show why every CTL formula can be transformed into normal form, as we have stated in the introduction.

2.2 Semantics

The semantics of CTL are defined in terms of temporal Kripke structures.

DEFINITION 2.3. *A **temporal Kripke structure** $\mathcal{K}$ for AP is a tuple $\langle W, R, s_0, \mathcal{V} \rangle$ where:*

- *W is the set of states,*
- *$R \subseteq W \times W$ is the **total**[1] accessibility relation (i.e., $\forall s \exists t R(s,t)$),*
- *$s_0 \in W$ is the initial state, and*
- *$\mathcal{V} : W \longrightarrow 2^{AP}$ is a valuation that determines which atomic propositions are true at each state.*

DEFINITION 2.4. *A **path** π of $\mathcal{K}$ is an infinite sequence $s_0, s_1, s_2, \ldots$ of states of W, such that $R(s_i, s_{i+1})$, $i \geq 0$. We also use the notational convention $\pi^i = s_i, s_{i+1}, s_{i+2}, \ldots$.*

$\mathcal{K}$ is, in general, a directed labelled graph with node set W, arc set R and labelling function $\mathcal{V}$. $\mathcal{K}$ need no be a tree; however, it can be viewed as an W-labelled tree, which results by **unwinding** $\mathcal{K}$ from the initial state s_0 (see [7] for details).

The truth value of CTL formulas is defined in terms of Kripke structures. The symbolism $\mathcal{K}, s \models \varphi$ means that "in structure $\mathcal{K}$ formula φ is true at state s" and the symbolism $\mathcal{K}, \pi \models \varphi$ means that "in structure $\mathcal{K}$ formula φ is true of path π".

DEFINITION 2.5. *We define recursively the meaning of $\models$. If φ is a state formula, then:*

- *$\models \top$ and $\not\models \bot$,*
- *$\mathcal{K}, s \models p \iff p \in \mathcal{V}(s)$, for an atomic proposition $p \in AP$,*
- *$\mathcal{K}, s \models \varphi \vee \psi \iff \mathcal{K}, s \models \varphi$ or $\mathcal{K}, s \models \psi$,*
- *$\mathcal{K}, s \models \varphi \wedge \psi \iff \mathcal{K}, s \models \varphi$ and $\mathcal{K}, s \models \psi$,*
- *$\mathcal{K}, s \models \mathbf{E}\varphi \iff$ there exists a path $\pi = s_0, s_1, \ldots$, with initial state $s = s_0$, such that $\mathcal{K}, \pi \models \varphi$, and*
- *$\mathcal{K}, s \models \mathbf{A}\varphi \iff$ for every path $\pi = s_0, s_1, \ldots$, with initial state $s = s_0$ it holds that $\mathcal{K}, \pi \models \varphi$.*

If φ is a path formula, then:

- *$\mathcal{K}, \pi \models \bigcirc\varphi \iff \mathcal{K}, \pi^1 \models \varphi$,*
- *$\mathcal{K}, \pi \models \varphi \mathbf{U} \psi \iff$ there exists $i \geq 0$ such that $\mathcal{K}, \pi^i \models \psi$ and for all j, $0 \leq j < i$, $\mathcal{K}, \pi^j \models \varphi$, and*
- *$\mathcal{K}, \pi \models \varphi \widetilde{\mathbf{U}} \psi \iff$ for all $i \geq 0$ such that $\mathcal{K}, \pi^i \not\models \psi$ there exists j, $0 \leq j < i$, such that $\mathcal{K}, \pi^j \models \varphi$.*

From the above Definition 2.5 we can see that:

$$\pi \models \neg(\varphi \widetilde{\mathbf{U}} \psi) \Leftrightarrow \ \pi \models \neg\varphi \mathbf{U} \neg\psi \tag{1}$$

or, equivalently,

$$\pi \models \neg(\varphi \mathbf{U} \psi) \Leftrightarrow \pi \models \neg\varphi \widetilde{\mathbf{U}} \neg\psi \tag{2}$$

[1]The reason for this requirement is discussed in Section 4.

One can think of $\varphi\widetilde{\mathbf{U}}\psi$ as saying that "either ψ always holds along π, or the first occurrence of $\neg\psi$ is **strictly** preceded by an occurrence of φ". This modality is needed for transforming any CTL formula into an equivalent one in positive normal form. Actually, $\widetilde{\mathbf{U}}$ was introduced by Vardi in [24], [16] precisely for this reason. Hence, our assumption that every CTL formula is in positive normal form is justified.

3. DATALOG

DEFINITION 3.1 ([23]). *A **Datalog program** Π is a finite set of function-free Horn clauses, called **rules**, of the form:*

$$G(x_1, \ldots, x_n) \longleftarrow H_1(y_{1,1}, \ldots, y_{1,n_1}), \ldots, H_k(y_{k,1}, \ldots, y_{k,n_k})$$

where:

- *$x_1, \ldots, x_n$ are variables,*
- *$y_{i,j}$'s are either variables or constants,*
- *G is a predicate, called the **head** of the rule, and*
- *The $H_1(y_{1,1}, \ldots, y_{1,n_1}), \ldots, H_k(y_{k,1}, \ldots, y_{k,n_k})$ are predicates that make the **body** of the rule.*

- **IDB** (Intensional database predicates) are called the predicates which are defined by the logical rules. This means that they can appear either in the head of one or more rules or in the body.
- **EDB** (Extensional database predicates) are called the predicates whose relation is stored in the database. An EDB predicate can only appear the body of rules.
- A pair (Π, G) consisting of a Datalog program Π together with one of its IDB predicates G called **goal** predicate, defines a query.
- Datalog programs are built from atomic formulas, which are predicate symbols with a list of arguments, e.g., $P(A_1, \ldots, A_n)$, where P is a predicate symbol. Each predicate symbol is associated with a particular number of arguments that it takes.
- A Datalog program Π is said to be **monadic** if all the predicates occurring in the heads of the rules have arity one.
- For any database D over relations that correspond to the EDB predicates, $G(D)$ is the set of ground facts about G which can be deduced from D by applications of the rules in Π. The **bottom up evaluation** that we use later, is an algorithm that initializes the IDB predicates to be empty and repeatedly apply the rules to add tuples to the IDB predicates, until no new tuples can be added [23].
- Datalog$^{\neg}$ extends Datalog by allowing negation in the EDB predicates only.
- Datalog$^{\neg}_{Succ}$ extends Datalog$^{\neg}$ by allowing to use in the body an interpreted predicate $Succ(X, Y)$ meaning that Y is the successor of X (assuming that X, Y take values from a totally ordered domain). The semantics are the same. In this paper, we use the equivalent notation $X{+}1$ for $Succ$ and we also assume that we are given the number of elements in the domain denoted by $\mathbf{c}_{max}$.

4. FINITE KRIPKE STRUCTURES AS RELATIONAL DATABASES

In this section we explain how finite Kripke structures can be viewed as relational databases in a natural way and, therefore, can be studied in a Datalog setting.

DEFINITION 4.1. *Let AP be a finite set $\{p_0, \ldots, p_r\}$ of atomic propositions and assume that $\mathcal{K} = \langle W, R, s_0, \mathcal{V}\rangle$ is a finite Kripke structure for AP. To $\mathcal{K}$ corresponds the finite relational database $D_{\mathcal{K}} = \langle W, R, s_0, P_0, \ldots, P_r, \mathbf{TRUE}, \mathbf{FALSE}\rangle$ defined as follows:*

- *W is the set of states W,*
- *R is the accessibility relation R,*
- *s_0 is the initial state,*
- *$P_i = \{s \in W \mid p_i \in \mathcal{V}(s)\}$ contains the states at which p_i is true $(0 \leq i \leq r)$,*
- **TRUE** $= W$ (**TRUE** *corresponds to* $\top$*), and*
- **FALSE** $= \emptyset$ (**FALSE** *corresponds to* $\bot$*).*

Note: For convenience and clarity, in Section 6 (where we give the translation rules), we use predicates **TRUE** and **TRUE**. However, these can be easily removed as follows: a) The rules that contain predicate $False$ are deleted and b) The predicate $True$ is simply deleted and the rest of the rule is retained.

We must point out that in Kripke structures the set of states W can be **infinite**. Indeed Definition 2.3 is general enough to allow W to have any cardinality. However, in relational databases the universe W is **finite**. This is the reason why we have defined the above transformation only for finite Kripke structures. This restriction in turn poses another complication. In CTL we are dealing with **infinite** computation paths, which means that in order to ensure that the accessibility relation R is meaningful, we must ensure that R is **total** ([16]):

$$\forall x \exists y \; R(x, y) \tag{3}$$

Notice that every path $\pi = s_0, s_1, s_2, \ldots$ of $\mathcal{K}$ gives rise to the path $\pi = s_0, s_1, s_2, \ldots$, where $R(s_i, s_{i+1})$, $i \geq 0$, in $D_{\mathcal{K}}$.

Recall from section 2 that $\mathcal{K}$ has the form of a directed labelled graph. The fact that $\mathcal{K}$ is finite implies that every node of the graph has a **finite branching degree**. In other words for every $x \in W$ there exist k (for some $k \in N$ that, of course, depends on x) distinct elements $y_0, \ldots, y_{k-1}$ of W such that $R(x, y_0), \ldots, R(x, y_{k-1})$. We may assume that in the temporal Kripke structure $\mathcal{K}$ there is an implicit linear order among the successors (i.e., the children) of a node (viz., from the leftmost child to the right-most). This linear order among the successors of any state can be captured by introducing the relations $S_0, \ldots, S_{n-1}$ (n is the maximum branching degree of $\mathcal{K}$) in $D_{\mathcal{K}}$. These relations serve as a refinement of the "unordered" accessibility relation R, i.e., $R = \bigcup_{i=0}^{n-1} S_i$. Hence, for every node x with k successors we may write $S_0(x, y_0), \ldots, S_{k-1}(x, y_{k-1})$, instead of $R(x, y_0), \ldots, R(x, y_{k-1})$, meaning that $y_0, \ldots, y_{k-1}$ are the $first, \ldots, k^{th}$ child of x, respectively.

To simplify the translation of CTL formulas to Datalog programs, which will be presented in section 6, we shall assume that $\mathcal{K}$ has branching degree (at most) two and we shall use the relations S_0 and S_1 instead of R.

5. CTL FORMULAS VS. RELATIONAL QUERIES

In this section we explain how we view the Datalog programs that correspond to CTL formulas. State formulas are evaluated on a state; it is easy to see that this poses no problem for Datalog because Datalog predicates are evaluated on tuples. In this particular case the tuple is unary and corresponds to a state. For path formulas however we have to be more careful; path formulas are evaluated over a path, but a Datalog predicate cannot be evaluated on a path. However, CTL state formulas are built up from existentially or universally quantified path formulas. This reasoning leads us to not actually define a Datalog program for a path formula φ, but for two state formulas: one for $\mathbf{E}\varphi$ and one for $\mathbf{A}\varphi$.

Specifically, the correspondence between CTL formulas and Datalog programs is as follows:

- To every CTL state formula φ corresponds a Datalog program Π_φ with unary goal predicate G_φ.
- To every CTL path formula φ correspond two Datalog programs; the **existential** $\Pi^{\mathbf{E}}_\varphi$ with unary goal predicate $G^{\mathbf{E}}_\varphi$, and the **universal** $\Pi^{\mathbf{A}}_\varphi$ with unary goal predicate $G^{\mathbf{A}}_\varphi$.

The reason we construct two additional programs for every path formula φ is the following: $\Pi^{\mathbf{E}}_\varphi$ will be used in the construction of the Datalog program for the formula $\mathbf{E}\varphi$ and $\Pi^{\mathbf{A}}_\varphi$ will be used in the construction of the Datalog program for the formula $\mathbf{A}\varphi$. We shall prove that this translation maintains equivalence between CTL formulas and Datalog programs, in the following sense:
(1) If φ is a state formula, then:

$$\mathcal{K}, s \models \varphi \Leftrightarrow s \in G_\varphi(D_\mathcal{K}) \tag{4}$$

(2) If φ is a path formula, then there are two separate cases:

$$\left\{ \begin{array}{c} \mathcal{K}, \pi \models \varphi \text{ for a } \pi = s_0, \ldots \Leftrightarrow s_0 \in G^{\mathbf{E}}_\varphi(D_\mathcal{K}) \\ \mathcal{K}, \pi \models \varphi \text{ for every } \pi = s_0, \ldots \Leftrightarrow s_0 \in G^{\mathbf{A}}_\varphi(D_\mathcal{K}) \end{array} \right\} \tag{5}$$

6. TRANSLATING CTL INTO DATALOG

6.1 Translation rules

In this section, we show how to translate CTL formulas into Datalog programs for temporal Kripke structures with branching degree (at most) two. We view such Kripke structures as being described by the two relations S_0 and S_1; $S_0(x, y)$ ($S_1(x, y)$) expressing that y is the left (right) child of x[2]. In this translation we use Datalog with negation applied only to EDB predicates, and with the successor built-in predicates.

Notice that the relation $S_0(x, y)$ is total because of (3), which expresses the totality of R. However, the relation $S_1(x, y)$ may not be defined for every state x.

Let $\pi = s_0, s_1, s_2, \ldots$ be a path in $D_\mathcal{K}$; then:

$$S_0(s_i, s_{i+1}) \text{ or } S_1(s_i, s_{i+1}), \text{ for every } \ i \geq 0. \tag{6}$$

For each CTL state formula φ we construct a Datalog program Π_φ with unary goal predicate G_φ and for each CTL path formula φ we construct two Datalog programs, $\Pi^{\mathbf{E}}_\varphi$ with unary goal predicate $G^{\mathbf{E}}_\varphi$ and $\Pi^{\mathbf{A}}_\varphi$ with unary goal predicate $G^{\mathbf{A}}_\varphi$. This is done recursively as follows:

DEFINITION 6.1. *Let φ be a CTL state formula. Then:*
(1) If $\varphi \equiv p$ or $\varphi \equiv \neg p$, where $p \in AP$, or $\varphi \equiv \top$ or $\varphi \equiv \bot$, the corresponding programs are

- $\Pi_\varphi : G_\varphi(x) \longleftarrow P(x)$,
- $\Pi_\varphi : G_\varphi(x) \longleftarrow \neg P(x)$
- $\Pi_\varphi : G_\varphi(x) \longleftarrow \mathbf{TRUE}(x)$, *and*
- $\Pi_\varphi : G_\varphi(x) \longleftarrow \mathbf{FALSE}(x)$

(2) If $\varphi \equiv \psi_1 \vee \psi_2$ or $\varphi \equiv \psi_1 \wedge \psi_2$, then

- $\Pi_\varphi : \left\{ \begin{array}{l} G_\varphi(x) \longleftarrow G_{\psi_1}(x) \\ G_\varphi(x) \longleftarrow G_{\psi_2}(x) \\ \Pi_{\psi_1} \\ \Pi_{\psi_2} \end{array} \right.$
- $\Pi_\varphi : \left\{ \begin{array}{l} G_\varphi(x) \longleftarrow G_{\psi_1}(x), G_{\psi_2}(x) \\ \Pi_{\psi_1} \\ \Pi_{\psi_2} \end{array} \right.$

where $(\Pi_{\psi_1}, G_{\psi_1})$ and $(\Pi_{\psi_2}, G_{\psi_2})$ are the programs and goals that correspond to formulas ψ_1 and ψ_2 respectively.
(3) If $\varphi \equiv \mathbf{E}\psi$, where ψ is a path formula, the program is

- $\Pi_\varphi : \left\{ \begin{array}{l} G_\varphi(x) \longleftarrow G^{\mathbf{E}}_\psi(x) \\ \Pi^{\mathbf{E}}_\psi \end{array} \right.$

(4) If $\varphi \equiv \mathbf{A}\psi$, where ψ is a path formula, the program is

- $\Pi_\varphi : \left\{ \begin{array}{l} G_\varphi(x) \longleftarrow G^{\mathbf{A}}_\psi(x) \\ \Pi^{\mathbf{A}}_\psi \end{array} \right.$

DEFINITION 6.2. *Suppose now that φ is a path formula.*
(1) If $\varphi \equiv \bigcirc\psi$, where ψ is a state formula, then the existential and universal programs are

- $\Pi^{\mathbf{E}}_\varphi : \left\{ \begin{array}{l} G^{\mathbf{E}}_\varphi(x) \longleftarrow S_0(x, y), G_\psi(y) \\ G^{\mathbf{E}}_\varphi(x) \longleftarrow S_1(x, y), G_\psi(y) \\ \Pi_\psi \end{array} \right.$
- $\Pi^{\mathbf{A}}_\varphi : \left\{ \begin{array}{l} G^{\mathbf{A}}_\varphi(x) \longleftarrow S_0(x, y), \neg S_1(x, z), G_\psi(y) \\ G^{\mathbf{A}}_\varphi(x) \longleftarrow S_0(x, y), S_1(x, z), G_\psi(y), G_\psi(z) \\ \Pi_\psi \end{array} \right.$

(2) If $\varphi \equiv \psi_1 \mathbf{U} \psi_2$, where ψ_1 and ψ_2 are state formulas, the existential and universal programs are

- $\Pi^{\mathbf{E}}_\varphi : \left\{ \begin{array}{l} G^{\mathbf{E}}_\varphi(x) \longleftarrow G_{\psi_2}(x) \\ G^{\mathbf{E}}_\varphi(x) \longleftarrow G_{\psi_1}(x), S_0(x, y), G^{\mathbf{E}}_\varphi(y) \\ G^{\mathbf{E}}_\varphi(x) \longleftarrow G_{\psi_1}(x), S_1(x, y), G^{\mathbf{E}}_\varphi(y) \\ \Pi_{\psi_1} \\ \Pi_{\psi_2} \end{array} \right.$
- $\Pi^{\mathbf{A}}_\varphi : \left\{ \begin{array}{l} G^{\mathbf{A}}_\varphi(x) \longleftarrow G_{\psi_2}(x) \\ G^{\mathbf{A}}_\varphi(x) \longleftarrow G_{\psi_1}(x), S_0(x, y), \neg S_1(x, z), G^{\mathbf{A}}_\varphi(y) \\ G^{\mathbf{A}}_\varphi(x) \longleftarrow G_{\psi_1}(x), S_0(x, y), S_1(x, z), G^{\mathbf{A}}_\varphi(y), \\ \qquad\qquad G^{\mathbf{A}}_\varphi(z) \\ \Pi_{\psi_1} \\ \Pi_{\psi_2} \end{array} \right.$

(3) If $\varphi \equiv \psi_1 \widetilde{\mathbf{U}} \psi_2$ (ψ_1 and ψ_2 are state formulas), the existential and universal programs are

[2]Our results can be extended to temporal Kripke structures of bounded branching degree.

- $\Pi^{\mathbf{E}}_{\varphi}$:
$$\begin{cases} G^{\mathbf{E}}_{\varphi}(x) \longleftarrow G_{\psi_1}(x), G_{\psi_2}(x) \\ G^{\mathbf{E}}_{\varphi}(x) \longleftarrow TC_{\psi_2}(x,x) \\ G^{\mathbf{E}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x), S_0(x,y), G^{\mathbf{E}}_{\varphi}(y) \\ G^{\mathbf{E}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x), S_1(x,y), G^{\mathbf{E}}_{\varphi}(y) \\ TC_{\psi_2}(x,y) \longleftarrow G_{\psi_2}(x), S_0(x,y), G_{\psi_2}(y) \\ TC_{\psi_2}(x,y) \longleftarrow G_{\psi_2}(x), S_1(x,y), G_{\psi_2}(y) \\ TC_{\psi_2}(x,z) \longleftarrow G_{\psi_2}(x), S_0(x,y), TC_{\psi_2}(y,z) \\ TC_{\psi_2}(x,y) \longleftarrow G_{\psi_2}(x), S_1(x,y), TC_{\psi_2}(y,z) \\ \Pi_{\psi_1} \\ \Pi_{\psi_2} \end{cases}$$

- $\Pi^{\mathbf{A}}_{\varphi}$:
$$\begin{cases} G^{\mathbf{A}}_{\varphi}(x) \longleftarrow G_{\psi_1}(x), G_{\psi_2}(x) \\ G^{\mathbf{A}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x), S_0(x,y), \neg S_1(x,z), G^{\mathbf{A}}_{\varphi}(y) \\ G^{\mathbf{A}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{\varphi}(y), G^{\mathbf{A}}_{\varphi}(z) \\ G^{\mathbf{A}}_{\varphi}(x) \longleftarrow PL_{\psi_2}(x, \mathbf{c}_{max}) \\ PL_{\psi_2}(x,n) \longleftarrow G_{\psi_2}(x), S_0(x,y), \neg S_1(x,z), PL_{\psi_2}(y,n-1), n \leq \mathbf{c}_{max} \\ PL_{\psi_2}(x,n) \longleftarrow G_{\psi_2}(x), S_0(x,y), S_1(x,z), PL_{\psi_2}(y,n-1), PL_{\psi_2}(z,n-1), n \leq \mathbf{c}_{max} \\ PL_{\psi_2}(x,n) \longleftarrow G_{\psi_2}(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{\varphi}(y), PL_{\psi_2}(z,n-1), n \leq \mathbf{c}_{max} \\ PL_{\psi_2}(x,n) \longleftarrow G_{\psi_2}(x), S_0(x,y), S_1(x,z), PL_{\psi_2}(y,n-1), G^{\mathbf{A}}_{\varphi}(z), n \leq \mathbf{c}_{max} \\ PL_{\psi_2}(x,1) \longleftarrow G_{\psi_2}(x), S_0(x,y), \neg S_1(x,z), G_{\psi_2}(y) \\ PL_{\psi_2}(x,1) \longleftarrow G_{\psi_2}(x), S_0(x,y), S_1(x,z), G_{\psi_2}(y), G_{\psi_2}(z) \\ \Pi_{\psi_1} \\ \Pi_{\psi_2} \end{cases}$$

In the program $\Pi^{\mathbf{A}}_{\varphi}$, the constant $\mathbf{c}_{max}$ is equal to the cardinality $|W|$ of the temporal Kripke structure $\mathcal{K}$.

6.2 Useful facts

DEFINITION 6.3. *Let $\mathcal{K} = \langle W, R, s_0, \mathcal{V}\rangle$ be a Kripke structure with branching degree at most two, let $\varphi \equiv \psi_1 \mathbf{U} \psi_2$ and let $\mathcal{K}, \pi \models \varphi$ for every path π with initial state s (possibly $s \neq s_0$). We define the $\mathbf{U}$ **unwinding** of $\mathcal{K}$ from s, denoted $\mathcal{T}^{\varphi}_{s}$, as the least subset of $W \times N$ for which the following hold:*

1. *$(s,0) \in \mathcal{T}^{\varphi}_{s}$, and*
2. *If $(s,n) \in \mathcal{T}^{\varphi}_{s}$, $R(s,t)$, $\mathcal{K}, s \models \psi_1 \wedge \neg\psi_2$ and $\mathcal{K}, t \models \psi_1 \vee \psi_2$, then $(t, n+1) \in \mathcal{T}^{\varphi}_{s}$.*

DEFINITION 6.4. *Let $\mathcal{K} = \langle W, R, s_0, \mathcal{V}\rangle$ be a finite Kripke structure with branching degree at most two, let $\varphi \equiv \psi_1 \widetilde{\mathbf{U}} \psi_2$ and let $\mathcal{K}, \pi \models \varphi$ for every path π with initial state s (possibly $s \neq s_0$). We define the $\widetilde{\mathbf{U}}$ **unwinding** of $\mathcal{K}$ from s, denoted $\mathcal{T}^{\varphi}_{s}$, as the least subset of $W \times N$ for which the following hold:*

1. *$(s,0) \in \mathcal{T}^{\varphi}_{s}$, and*
2. *If $(s,n) \in \mathcal{T}^{\varphi}_{s}$, $n < |W| - 1$, $R(s,t)$, $\mathcal{K}, s \models \psi_2 \wedge \neg\psi_1$ and $\mathcal{K}, t \models \psi_2$, then $(t, n+1) \in \mathcal{T}^{\varphi}_{s}$.*

A trivial consequence of the fact that our finite Kripke structures have branching degree two is that every state in $D_{\mathcal{K}}$ has either a left child but not a right child, or both a left and a right child. We fix the following notation: given a state s_0 of $D_{\mathcal{K}}$, its left child will be denoted s^L_1 and its right child (if it exists) will be denoted s^R_1.

We state now without proof four easy lemmata that we are going to use extensively in the proof of the main result.

LEMMA 6.1. *Let $\mathcal{K} = \langle W, R, s_0, \mathcal{V}\rangle$ be a finite Kripke structure and let $s_0, \ldots, s_i, \ldots, s_n$ be a finite path in $\mathcal{K}$ (or in $D_{\mathcal{K}}$), where $|W| = n$. Then, there exists a state s such that $s_i = s_j = s$.* ⊣

LEMMA 6.2. *Let $\mathcal{K} = \langle W, R, s_0, \mathcal{V}\rangle$ be a Kripke structure with branching degree at most two, let $\varphi \equiv \psi_1 \mathbf{U} \psi_2$, let $\mathcal{K}, \pi \models \varphi$ for every path π with initial state s_0 and let $\mathcal{T}^{\varphi}_{s}$ be the $\mathbf{U}$ unwinding of $\mathcal{K}$ from s_0. Then the following hold:*

1. *$\mathcal{T}^{\varphi}_{s}$ is a finite tree with branching degree at most two.*
2. *If (s,n) is a leaf of $\mathcal{T}^{\varphi}_{s}$, then $\mathcal{K}, s \models \psi_2$.*
3. *If (s,n) is an internal node of $\mathcal{T}^{\varphi}_{s}$, then $\mathcal{K}, s \models \psi_1 \wedge \neg\psi_2$.* ⊣

LEMMA 6.3. *Let $\mathcal{K} = \langle W, R, s_0, \mathcal{V}\rangle$ be a finite Kripke structure with branching degree at most two, let $\varphi \equiv \psi_1 \widetilde{\mathbf{U}} \psi_2$, let $\mathcal{K}, \pi \models \varphi$ for every path π with initial state s_0 and let $\mathcal{T}^{\varphi}_{s}$ be the $\widetilde{\mathbf{U}}$ unwinding of $\mathcal{K}$ from s_0. The following hold:*

1. *$\mathcal{T}^{\varphi}_{s}$ is a finite tree of height at most $|W| - 1$ that has branching degree at most two.*
2. *If (s,n) is a leaf of $\mathcal{T}^{\varphi}_{s}$, then either*
   (a) *$\mathcal{K}, s \models \psi_1 \wedge \psi_2$, or*
   (b) *$n = |W| - 1$, $\mathcal{K}, s \models \neg\psi_1 \wedge \psi_2$ and for every child t of s (in $D_{\mathcal{K}}$) $\mathcal{K}, t \models \psi_2$.*
3. *If (s,n) is an internal node of $\mathcal{T}^{\varphi}_{s}$, then $n < |W| - 1$ and $\mathcal{K}, s \models \neg\psi_1 \wedge \psi_2$.* ⊣

The IDB predicate $TC_{\psi_2}(x,y)$ of Definition 6.2 captures the notion of a path from state x to state y, such that ψ_2 holds at every state along this path. Therefore, $TC_{\psi_2}(x,x)$ asserts the existence of a cycle with the property that ψ_2 holds at every state of this cycle. All these are formalized in the following Lemma.

LEMMA 6.4.

1. *Let $s_0, \ldots, s_n$ be a finite path in $D_{\mathcal{K}}$ such that $s_0 = s_n = s$ and $G_{\psi_2}(s_i)$, for every i, $0 \leq i \leq n$. Then predicate $TC_{\psi_2}(s,s)$ is true.*
2. *Let $TC_{\psi_2}(s,s)$ hold. Then, there exists a finite path $s_0, \ldots, s_n$ in $D_{\mathcal{K}}$ such that $s_0 = s_n = s$ and $G_{\psi_2}(s_i)$, for every i, $0 \leq i \leq n$.* ⊣

6.3 The main result

We are ready now to prove the main result.

THEOREM 6.1. *Let AP be a finite set of atomic propositions, let $\mathcal{K} = \langle W, R, s_0, \mathcal{V}\rangle$ be a finite Kripke structure for AP and let $D_{\mathcal{K}}$ be the corresponding relational database. Then, the following hold:*
(1) If φ is a state formula

$$\mathcal{K}, s \models \varphi \Leftrightarrow s \in G_{\varphi}(D_{\mathcal{K}}) \tag{7}$$

(2) If φ is a path formula

$$\left\{ \begin{array}{c} \mathcal{K}, \pi \models \varphi \text{ for a } \pi = s_0, \ldots \Leftrightarrow s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}}) \\ \mathcal{K}, \pi \models \varphi \text{ for every } \pi = s_0, \ldots \Leftrightarrow s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}) \end{array} \right\} \quad (8)$$

We shall prove that (7) and (8) hold by simultaneous induction on the structure of the formula φ. To make the proof more readable we consider the two directions separately. We begin by considering the $\Rightarrow$ direction.

Proof ($\Rightarrow$)

Let us consider a state formula φ. We must examine the following cases:

1. If $\varphi \equiv p$ or $\varphi \equiv \neg p$, where $p \in AP$, then the corresponding programs are those of Definition 6.1.(1). Trivially, then:
   - $\mathcal{K}, s \models p \Rightarrow p \in V(s) \Rightarrow P(s)$ is a ground fact of $D_{\mathcal{K}} \Rightarrow s \in G_{\varphi}(D_{\mathcal{K}})$, and
   - $\mathcal{K}, s \models \neg p \Rightarrow p \notin V(s) \Rightarrow P(s)$ is not a ground fact of $D_{\mathcal{K}} \Rightarrow s \in G_{\varphi}(D_{\mathcal{K}})$
2. If $\varphi \equiv \psi_1 \vee \psi_2$ or $\varphi \equiv \psi_1 \wedge \psi_2$, then the corresponding programs are shown in Definition 6.1.(2). Again, the following hold:
   - $\mathcal{K}, s \models \varphi \Rightarrow \mathcal{K}, s \models \psi_1$ or $\mathcal{K}, s \models \psi_2 \Rightarrow$ (by the inductive hypothesis) $s \in G_{\psi_1}(D_{\mathcal{K}})$ or $s \in G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow s \in G_{\psi_1}(D_{\mathcal{K}}) \cup G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow s \in G_{\varphi}(D_{\mathcal{K}})$
   - $\mathcal{K}, s \models \varphi \Rightarrow \mathcal{K}, s \models \psi_1$ and $\mathcal{K}, s \models \psi_2 \Rightarrow$ (by the inductive hypothesis) $s \in G_{\psi_1}(D_{\mathcal{K}})$ and $s \in G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow s \in G_{\psi_1}(D_{\mathcal{K}}) \cap G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow s \in G_{\varphi}(D_{\mathcal{K}})$
3. If $\varphi \equiv \mathbf{E}\psi$, where ψ is a path formula, then the corresponding program is that of Definition 6.1.(3).
   - $\mathcal{K}, s \models \mathbf{E}\psi \Rightarrow$ there exists a path $\pi = s_0, s_1, \ldots$ with initial state $s_0 = s$, such that $\mathcal{K}, \pi \models \psi \Rightarrow$ (by the inductive hypothesis) $s \in G^{\mathbf{E}}_{\psi}(D_{\mathcal{K}}) \Rightarrow s \in G_{\varphi}(D_{\mathcal{K}})$
4. If $\varphi \equiv \mathbf{A}\psi$, where ψ is a path formula, then the corresponding program is that of Definition 6.1.(4).
   - $\mathcal{K}, s \models \mathbf{A}\psi \Rightarrow$ for every path $\pi = s_0, s_1, \ldots$ with initial state $s_0 = s$, we have $\mathcal{K}, \pi \models \psi \Rightarrow$ (by the inductive hypothesis) $s \in G^{\mathbf{A}}_{\psi}(D_{\mathcal{K}}) \Rightarrow s \in G_{\varphi}(D_{\mathcal{K}})$

Let us consider now the more interesting case of path formulas; let φ be a CTL path formula.

1. If $\varphi \equiv \bigcirc\psi$, where ψ is a state formula, then the corresponding existential and universal programs are shown in Definition 6.2.(1).
   - The existential case:
     Let us assume that $\mathcal{K}, \pi \models \varphi$ for some path $\pi = s_0, s_1, s_2, \ldots$ with initial state s_0. From (6) we know that either $S_0(s_0, s_1)$ or $S_1(s_0, s_1)$ holds. Now $\mathcal{K}, \pi \models \varphi$ for the path $\pi = s_0, s_1, s_2, \ldots \Rightarrow \mathcal{K}, \pi^1 \models \psi$ for the path $\pi^1 = s_1, s_2, \ldots \Rightarrow \mathcal{K}, s_1 \models \psi$ (because ψ is a state formula) $\Rightarrow$ (by the inductive hypothesis) $s_1 \in G_{\psi}(D_{\mathcal{K}})$. From $\Pi^{\mathbf{E}}_{\varphi}$, by combining $G_{\psi}(s_1)$ with one of $S_0(s_0, s_1)$ or $S_1(s_0, s_1)$, we immediately derive $G^{\mathbf{E}}_{\varphi}(s_0)$ and, thus, $s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}})$.
   - The universal case:
     Let's assume now that $\mathcal{K}, \pi \models \varphi$ for every path $\pi = s_0, s_1, s_2, ...$ with initial state s_0. It is convenient to distinguish two cases:
     (a) s_0 has a left child s^L_1, but not a right child. In this case $\mathcal{K}, \pi \models \varphi$ for every path $\pi = s_0, s_1, s_2, \ldots$ with initial state $s_0 \Rightarrow \mathcal{K}, \pi^{1,L} \models \psi$ for every path $\pi^{1,L} = s^L_1, s_2, \ldots$ with initial state $s^L_1 \Rightarrow \mathcal{K}, s^L_1 \models \psi$ (because ψ is a state formula) $\Rightarrow$ (by the inductive hypothesis) $s^L_1 \in G_{\psi}(D_{\mathcal{K}})$. Moreover, in this case $S_0(s_0, s^L_1)$, $\neg S_1(s_0, x)$ are true and, therefore, evaluation of the first rule of $\Pi^{\mathbf{A}}_{\varphi}$ gives $G^{\mathbf{A}}_{\varphi}(s_0) \longleftarrow S_0(s_0, s^L_1), \neg S_1(s_0, x), G_{\psi}(s^L_1) \Rightarrow s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$.
     (b) s_0 has both a left child s^L_1 and a right child s^R_1. Then $\mathcal{K}, \pi \models \varphi$ for every path $\pi = s_0, s_1, s_2, \ldots$ with initial state $s_0 \Rightarrow \mathcal{K}, \pi^{1,L} \models \psi$ for every path $\pi^{1,L} = s^L_1, s^L_2, \ldots$ with initial state s^L_1 and $\mathcal{K}, \pi^{1,R} \models \psi$ for every path $\pi^{1,R} = s^R_1, s^R_2, \ldots$ with initial state $s^R_1 \Rightarrow \mathcal{K}, s^L_1 \models \psi$ and $\mathcal{K}, s^R_1 \models \psi$ (because ψ is a state formula) $\Rightarrow$ (by the inductive hypothesis) $s^L_1, s^R_1 \in G_{\psi}(D_{\mathcal{K}})$. Moreover, in this case $S_0(s_0, s^L_1), S_1(s_0, s^R_1)$ are true and, therefore, evaluation of the second rule of $\Pi^{\mathbf{A}}_{\varphi}$ gives $G^{\mathbf{A}}_{\varphi}(s_0) \longleftarrow S_0(s_0, s^L_1)$, $S_1(s_0, s^R_1)$, $G_{\psi}(s^L_1)$, $G_{\psi}(s^R_1) \Rightarrow s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$.
2. If $\varphi \equiv \psi_1 \mathbf{U} \psi_2$, where ψ_1 and ψ_2 are state formulas, then the corresponding existential and universal programs are those of Definition 6.2.(2).
   - The existential case:
     Suppose that $\mathcal{K}, \pi \models \varphi$ where the path π is $s_0, s_1, s_2, \ldots$. We have to examine two cases:
     (a) $\mathcal{K}, \pi \models \psi_2$ for the path $\pi = s_0, s_1, s_2, \ldots \Rightarrow \mathcal{K}, s_0 \models \psi_2$ (because ψ_2 is a state formula) $\Rightarrow$ (by the inductive hypothesis) $s_0 \in G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow G^{\mathbf{E}}_{\varphi}(s_0)$ (from the first rule $G^{\mathbf{E}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x)$ of $\Pi^{\mathbf{E}}_{\varphi}$) $\Rightarrow s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}})$.
     (b) $\mathcal{K}, \pi^i \models \psi_2$ for the path $\pi^i = s_i, s_{i+1}, s_{i+2}, \ldots$ and $\mathcal{K}, \pi^j \models \psi_1$ for $\pi^j = s_j, s_{j+1}, s_{j+2}, \ldots$ $(0 \leq j \leq i-1) \Rightarrow \mathcal{K}, s_i \models \psi_2$ and $\mathcal{K}, s_j \models \psi_1$ $(0 \leq j \leq i-1)$ (because ψ_1 and ψ_2 are state formulas) $\Rightarrow s_i \in G_{\psi_2}(D_{\mathcal{K}})$ and $s_j \in G_{\psi_1}(D_{\mathcal{K}})$ $(0 \leq j \leq i-1)$ (by the inductive hypothesis). From (6) we know that for every r, $0 \leq r < i$, at least one of $S_0(s_r, s_{r+1})$ or $S_1(s_r, s_{r+1})$ holds. From the first rule $G^{\mathbf{E}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x)$ of $\Pi^{\mathbf{E}}_{\varphi}$ we derive that $G^{\mathbf{E}}_{\varphi}(s_i)$. Successive applications of the second $(G^{\mathbf{E}}_{\varphi}(s_r) \longleftarrow G_{\psi_1}(s_r), S_0(s_r, s_{r+1}), G^{\mathbf{E}}_{\varphi}(s_{r+1}))$ and third rule $(G^{\mathbf{E}}_{\varphi}(s_r) \longleftarrow G_{\psi_1}(s_r), S_1(s_r, s_{r+1}), G^{\mathbf{E}}_{\varphi}(s_{r+1}))$ of $\Pi^{\mathbf{E}}_{\varphi}$ for every r, $0 \leq r < i$, yield $G^{\mathbf{E}}_{\varphi}(s_{i-1}), G^{\mathbf{E}}_{\varphi}(s_{i-2}), \ldots, G^{\mathbf{E}}_{\varphi}(s_1), G^{\mathbf{E}}_{\varphi}(s_0)$. Thus, $s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}})$.
   - The universal case:
     Let us assume now that $\mathcal{K}, \pi \models \varphi$ for every path $\pi = s_0, s_1, s_2, \ldots$ with initial state s_0. Consider the $\mathbf{U}$ unwinding $\mathcal{T}^{\varphi}_{s_0}$ of $\mathcal{K}$ from s_0 and let (t_0, r) be any node of $\mathcal{T}^{\varphi}_{s_0}$; we shall prove that $t_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$. This property of $\mathcal{T}^{\varphi}_{s_0}$ indeed implies the required result because $(s_0, 0)$ is a node (specifically the root) of $\mathcal{T}^{\varphi}_{s_0}$ and, thus, $s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$. To

prove it, let $L_{t_0} = (t_0, r), (t_1, r+1), \ldots, (t_n, r+n)$ be the longest path from (t_0, r) to a leaf $(t_n, r+n)$ of $\mathcal{T}^{\varphi}_{s_0}$. We use induction on the length n of the path L_{t_0}.
(a) If $n = 0$, then node (t_0, r) itself is a leaf. From Lemma 6.2 we know that $\mathcal{K}, t_0 \models \psi_2$ and by the inductive hypothesis (pertaining to formula ψ_2) we get that $t_0 \in G_{\psi_2}(D_{\mathcal{K}})$. Then, from rule $G^{\mathbf{A}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x)$ of $\Pi^{\mathbf{A}}_{\varphi}$, we derive that $G^{\mathbf{A}}_{\varphi}(t_0)$.
(b) We show now that the claim holds for paths of length $n+1$, assuming that it holds for paths of length n. In this case node (t_0, r) is an internal node of $\mathcal{T}^{\varphi}_{s_0}$. From Lemma 6.2 we know that $\mathcal{K}, t_0 \models \psi_1$ and by the inductive hypothesis (pertaining to formula ψ_1) we get that $t_0 \in G_{\psi_1}(D_{\mathcal{K}})$. We focus on the case where node (t_0, r) has exactly two successors $(t^L_1, r+1)$ and $(t^R_1, r+1)$ in $\mathcal{T}^{\varphi}_{s_0}$ (the case where (t_0, r) has only one successor is identical). Since L_{t_0} has length $n+1$, then both $L_{t^L_1}$ and $L_{t^R_1}$ have length at most n. Hence, by the inductive hypothesis with respect to the length of the paths $L_{t^L_1}$ and $L_{t^R_1}$, we get that t^L_1 and $t^R_1 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$. So $G_{\psi_1}(t_0)$, $S_0(t_0, t^L_1)$, $S_1(t_0, t^R_1)$, $G^{\mathbf{A}}_{\varphi}(t^L_1)$ and $G^{\mathbf{A}}_{\varphi}(t^R_1)$ are true and, therefore, the third rule of $\Pi^{\mathbf{A}}_{\varphi}$ gives that $G^{\mathbf{A}}_{\varphi}(t_0) \longleftarrow G_{\psi_1}(t_0)$, $S_0(t_0, t^L_1)$, $S_1(t_0, t^R_1)$, $G^{\mathbf{A}}_{\varphi}(t^L_1)$, $G^{\mathbf{A}}_{\varphi}(t^R_1)$. Thus, $t_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$.

3. If $\varphi \equiv \psi_1 \widetilde{\mathbf{U}} \psi_2$, where ψ_1 and ψ_2 are state formulas, then the corresponding existential and universal programs are shown in Definition 6.2.(3).

   - The existential case:
     Suppose that $\mathcal{K}, \pi \models \varphi$ where the path π is $s_0, s_1, s_2, \ldots$. We must consider two cases:
     (a) $\mathcal{K}, \pi^i \models \psi_1 \wedge \psi_2$ for the path $\pi^i = s_i, s_{i+1}, s_{i+2}, \ldots$ and $\mathcal{K}, \pi^j \models \psi_2$ for $\pi^j = s_j, s_{j+1}, s_{j+2}, \ldots$ $(0 \leq j \leq i-1) \Rightarrow \mathcal{K}, s_i \models \psi_1 \wedge \psi_2$ and $\mathcal{K}, s_j \models \psi_2$ $(0 \leq j \leq i-1)$ (because ψ_1 and ψ_2 are state formulas) $\Rightarrow s_i \in G_{\psi_1}(D_{\mathcal{K}})$ and $s_j \in G_{\psi_2}(D_{\mathcal{K}})$ $(0 \leq j \leq i)$ (by the inductive hypothesis). From (6) we know that for every r, $0 \leq r < i$, at least one of $S_0(s_r, s_{r+1})$ or $S_1(s_r, s_{r+1})$ holds. From rule $G^{\mathbf{E}}_{\varphi}(x) \longleftarrow G_{\psi_1}(x), G_{\psi_2}(x)$ of $\Pi^{\mathbf{E}}_{\varphi}$ we derive that $G^{\mathbf{E}}_{\varphi}(s_i)$. Successive applications of the other two rules of the program $\Pi^{\mathbf{E}}_{\varphi}$ (i.e., $G^{\mathbf{E}}_{\varphi}(s_r) \longleftarrow G_{\psi_2}(s_r)$, $S_0(s_r, s_{r+1})$, $G^{\mathbf{E}}_{\varphi}(s_{r+1})$ and $G^{\mathbf{E}}_{\varphi}(s_r) \longleftarrow G_{\psi_2}(s_r)$, $S_1(s_r, s_{r+1})$, $G^{\mathbf{E}}_{\varphi}(s_{r+1})$) for every r, $0 \leq r < i$, yield $G^{\mathbf{E}}_{\varphi}(s_{i-1})$, $G^{\mathbf{E}}_{\varphi}(s_{i-2}), \ldots, G^{\mathbf{E}}_{\varphi}(s_1)$, $G^{\mathbf{E}}_{\varphi}(s_0)$. Thus, $s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}})$.
     (b) $\mathcal{K}, \pi^i \models \psi_2$ for the path $\pi^i = s_i, s_{i+1}, s_{i+2}, \ldots$, for every $i \geq 0$. This implies that $\mathcal{K}, s_i \models \psi_2$, for every $i \geq 0$, and (by the inductive hypothesis) that $s_i \in G_{\psi_2}(D_{\mathcal{K}})$, for every $i \geq 0$. Let $s_0, s_1, s_2, \ldots, s_n$ be an initial segment of π, where $n = |W|$. From Lemma 6.1 we know that in the aforementioned sequence there exists a state s such that $s = s_k = s_l$, $0 \leq k < l \leq n$. Then Lemma 6.4.(1) implies that $s_k \in TC_{\psi_2}(D_{\mathcal{K}})$. From (6) we know that for every r, $0 \leq r < k$, at least one of $S_0(s_r, s_{r+1})$ or $S_1(s_r, s_{r+1})$ holds. From rule $G^{\mathbf{E}}_{\varphi}(x) \longleftarrow TC_{\psi_2}(x, x)$ we derive that $G^{\mathbf{E}}_{\varphi}(s_k)$. Successive applications of the third $(G^{\mathbf{E}}_{\varphi}(s_r) \longleftarrow G_{\psi_2}(s_r),\ S_0(s_r, s_{r+1}),\ G^{\mathbf{E}}_{\varphi}(s_{r+1}))$ or the fourth $(G^{\mathbf{E}}_{\varphi}(s_r) \longleftarrow G_{\psi_2}(s_r), S_1(s_r, s_{r+1}), G^{\mathbf{E}}_{\varphi}(s_{r+1}))$ rule of $\Pi^{\mathbf{E}}_{\varphi}$ for every r, $0 \leq r < k$, yield $G^{\mathbf{E}}_{\varphi}(s_{i-1})$, $G^{\mathbf{E}}_{\varphi}(s_{i-2}), \ldots, G^{\mathbf{E}}_{\varphi}(s_1), G^{\mathbf{E}}_{\varphi}(s_0)$. Accordingly, $s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}})$.

   - The universal case:
     Let us assume now that $\mathcal{K}, \pi \models \varphi$ for every path $\pi = s_0, s_1, s_2, \ldots$ with initial state s_0. Consider the $\widetilde{\mathbf{U}}$ unwinding $\mathcal{T}^{\varphi}_{s_0}$ of $\mathcal{K}$ from s_0 and let (t_0, r) be any node of $\mathcal{T}^{\varphi}_{s_0}$. We shall prove that either $G^{\mathbf{A}}_{\varphi}(t_0)$ or $PL_{\psi_2}(t_0, |W| - r)$ hold. This property of the nodes of $\mathcal{T}^{\varphi}_{s_0}$ ensures that for the root $(s_0, 0)$ it must be the case that $s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$ (recall the universal program in Definition 6.2.(3)). To prove this property, let $L_{t_0} = (t_0, r), (t_1, r+1), \ldots, (t_n, r+n)$ be the longest path from (t_0, r) to a leaf $(t_n, r+n)$ of $\mathcal{T}^{\varphi}_{s_0}$. We are going to use induction on the length n of the path L_{t_0}.
     (a) If $n = 0$, then node (t_0, r) itself is a leaf. We may assume that node t_0 has exactly two successors t^L_1 and t^R_1 in $D_{\mathcal{K}}$ because the case where t_0 has only one successor can be tackled in the same way. From Lemma 6.3 we know that there are two cases regarding t_0: (1) $\mathcal{K}, t_0 \models \psi_1 \wedge \psi_2$; in this case the inductive hypothesis (with respect to ψ_1 and ψ_2) gives that $t_0 \in G_{\psi_1}(D_{\mathcal{K}})$ and $t_0 \in G_{\psi_2}(D_{\mathcal{K}})$ and from the first rule of $\Pi^{\mathbf{A}}_{\varphi}$ we derive that $G^{\mathbf{A}}_{\varphi}(t_0)$. (2) $r = |W| - 1$, $\mathcal{K}, t_0 \models \neg\psi_1 \wedge \psi_2$ and $\mathcal{K}, t^L_1 \models \psi_2$ and $\mathcal{K}, t^R_1 \models \psi_2$. The inductive hypothesis with respect to ψ_2 gives that $t_0 \in G_{\psi_2}(D_{\mathcal{K}})$, $t^L_1 \in G_{\psi_2}(D_{\mathcal{K}})$ and $t^R_1 \in G_{\psi_2}(D_{\mathcal{K}})$. Using rule $PL_{\psi_2}(t_0, 1) \longleftarrow G_{\psi_2}(t_0)$, $S_0(t_0, t^L_1)$, $S_1(t_0, t^R_1)$, $G_{\psi_2}(t^L_1)$, $G_{\psi_2}(t^R_1)$ of $\Pi^{\mathbf{A}}_{\varphi}$ we conclude that $PL_{\psi_2}(t_0, 1)$ holds.
     (b) We prove now that the claim holds for paths of length $n+1$, assuming that it holds for paths of length n. In this case node (t_0, r) is an internal node of $\mathcal{T}^{\varphi}_{s_0}$. ¿From Lemma 6.3 we know that $r < |W| - 1$ and $\mathcal{K}, t_0 \models \neg\psi_1 \wedge \psi_2$ and by the inductive hypothesis (pertaining to ψ_2) we get that $t_0 \in G_{\psi_2}(D_{\mathcal{K}})$. We examine the case where node (t_0, r) has exactly two successors $(t^L_1, r+1)$ and $(t^R_1, r+1)$ in $\mathcal{T}^{\varphi}_{s_0}$, where $r + 1 < |W|$. In this case $S_0(t_0, t^L_1)$, $S_1(t_0, t^R_1)$ are true. Since L_{t_0} has length $n+1$, then both $L_{t^L_1}$ and $L_{t^R_1}$ have length at most n. Hence, by the inductive hypothesis (regarding the path length), we get that $G^{\mathbf{A}}_{\varphi}(t^L_1)$ or $PL_{\psi_2}(t^L_1, |W| - r - 1)$ and $G^{\mathbf{A}}_{\varphi}(t^R_1)$ or $PL_{\psi_2}(t^R_1, |W| - r - 1)$. If $G^{\mathbf{A}}_{\varphi}(t^L_1)$ and $G^{\mathbf{A}}_{\varphi}(t^R_1)$ are true, then the third rule of $\Pi^{\mathbf{A}}_{\varphi}$ $(G^{\mathbf{A}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x), S_0(x, y), S_1(x, z), G^{\mathbf{A}}_{\varphi}(y), G^{\mathbf{A}}_{\varphi}(z))$ implies that $G^{\mathbf{A}}_{\varphi}(t_0)$ also holds. If $PL_{\psi_2}(t^L_1, |W| - r - 1)$ and $PL_{\psi_2}(t^R_1, |W| - r - 1)$ are true, then using the sixth rule of $\Pi^{\mathbf{A}}_{\varphi}$ $(PL_{\psi_2}(x, n) \longleftarrow G_{\psi_2}(x), S_0(x, y), S_1(x, z), PL_{\psi_2}(y, n-1), PL_{\psi_2}(z, n-1), n \leq |W|)$ we conclude that $PL_{\psi_2}(t_0, |W| - r)$ also holds. In the remaining two cases the seventh and eighth rule imply that $PL_{\psi_2}(t_0, |W| - r)$.
     We have, thus, proved that for the node $(s_0, 0)$ one of $G^{\mathbf{A}}_{\varphi}(s_0)$ or $PL_{\psi_2}(s_0, |W|)$ holds. If we as-

sume that $PL_{\psi_2}(s_0, |W|)$ holds, then the fourth rule of $\Pi^{\mathbf{A}}_{\varphi}$ implies that $G^{\mathbf{A}}_{\varphi}(s_0)$. Hence, in any case, $s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$. $\dashv$

We complete now the proof of (7) and (8) by considering the opposite direction.

Proof ($\Leftarrow$)

First we consider the case where φ is a state formula.

1. If $\varphi \equiv p$ or $\varphi \equiv \neg p$, where $p \in AP$, then the corresponding programs are those of Definition 6.1.(1). Trivially, then:
   - $s \in G_{\varphi}(D_{\mathcal{K}}) \Rightarrow P(s)$ is a ground fact of $D_{\mathcal{K}} \Rightarrow p \in V(s) \Rightarrow \mathcal{K}, s \models p$, and
   - $s \in G_{\varphi}(D_{\mathcal{K}}) \Rightarrow P(s)$ is not a ground fact of $D_{\mathcal{K}} \Rightarrow p \notin V(s) \Rightarrow \mathcal{K}, s \models \neg p$
2. If $\varphi \equiv \psi_1 \vee \psi_2$ or $\varphi \equiv \psi_1 \wedge \psi_2$, then the corresponding programs are shown in Definition 6.1.(2). Again, the following hold:
   - $s \in G_{\varphi}(D_{\mathcal{K}}) \Rightarrow s \in G_{\psi_1}(D_{\mathcal{K}}) \cup G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow s \in G_{\psi_1}(D_{\mathcal{K}})$ or $s \in G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow$ (by the inductive hypothesis) $\mathcal{K}, s \models \psi_1$ or $\mathcal{K}, s \models \psi_2 \Rightarrow \mathcal{K}, s \models \varphi$.
   - $s \in G_{\varphi}(D_{\mathcal{K}}) \Rightarrow s \in G_{\psi_1}(D_{\mathcal{K}}) \cap G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow s \in G_{\psi_1}(D_{\mathcal{K}})$ and $s \in G_{\psi_2}(D_{\mathcal{K}}) \Rightarrow$ (by the inductive hypothesis) $\mathcal{K}, s \models \psi_1$ and $\mathcal{K}, s \models \psi_2 \Rightarrow \mathcal{K}, s \models \varphi$.
3. If $\varphi \equiv \mathbf{E}\psi$, where ψ is a path formula, then the corresponding program is depicted in Definition 6.1.(3).
   - $s \in G_{\varphi}(D_{\mathcal{K}}) \Rightarrow s \in G^{\mathbf{E}}_{\psi}(D_{\mathcal{K}}) \Rightarrow$ (by the inductive hypothesis) there exists a path $\pi = s_0, s_1, \ldots$ with initial state $s_0 = s$, such that $\mathcal{K}, \pi \models \psi \Rightarrow \mathcal{K}, s \models \mathbf{E}\psi$
4. If $\varphi \equiv \mathbf{A}\psi$, where ψ is a path formula, then the corresponding program that of Definition 6.1.(4).
   - $s \in G_{\varphi}(D_{\mathcal{K}}) \Rightarrow s \in G^{\mathbf{A}}_{\psi}(D_{\mathcal{K}}) \Rightarrow$ (by the inductive hypothesis) for every path $\pi = s_0, s_1, \ldots$ with initial state $s_0 = s$, we have $\mathcal{K}, \pi \models \psi \Rightarrow \mathcal{K}, s \models \mathbf{A}\psi$

Now we consider the case where φ is a CTL path formula.

1. If $\varphi \equiv \bigcirc\psi$, where ψ is a state formula, then the corresponding existential and universal programs are shown in 6.2.(1).
   - The existential case:
     Let us assume that $s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}})$. From the rules of the program $\Pi^{\mathbf{E}}_{\varphi}$ we see that there exists a s_1 such that $G_{\psi}(s_1)$ holds and also one of $S_0(s_0, s_1)$ or $S_1(s_0, s_1)$ holds. By the inductive hypothesis we get $\mathcal{K}, s_1 \models \psi$ (ψ is a state formula). Let $\pi = s_0, s_1, s_2, \ldots$ be any path with initial state s_0 and second state s_1. Clearly, then $\mathcal{K}, \pi^1 \models \psi$ for the path $\pi^1 = s_1, s_2, \ldots$ and $\mathcal{K}, \pi \models \varphi$ for the path $\pi = s_0, s_1, s_2, \ldots$.
   - The universal case:
     Suppose now that $s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$. It is convenient to distinguish two cases:
     (a) s_0 has a left successor s^L_1 but not a right successor in $D_{\mathcal{K}}$; in this case $S_0(s_0, s^L_1)$ and $\neg S_1(s_0, x)$ are true. From the first rule of $\Pi^{\mathbf{A}}_{\varphi}$ we see that $G_{\psi}(s^L_1)$ holds. By the inductive hypothesis we get $\mathcal{K}, s^L_1 \models \psi$ (ψ is a state formula). Let $\pi = s_0, s_1, s_2, \ldots$ be an arbitrary path with initial state s_0. The fact that s_0 has a left successor s^L_1 but not a right successor implies that s^L_1 is the second state of every such path. Suppose that there exists a path $\pi = s_0, s^L_1, s_2, \ldots$ with initial state s_0 such that $\mathcal{K}, \pi \not\models \varphi$. Trivially then $\mathcal{K}, \pi^1 \not\models \psi$, where $\pi^1 = s^L_1, s_2, \ldots$, which in turn implies that $\mathcal{K}, s^L_1 \not\models \psi$, which is absurd.
     (b) s_0 has both a left successor s^L_1 and a right successor s^R_1 in $D_{\mathcal{K}}$; in this case $S_0(s_0, s^L_1)$ and $S_1(s_0, s^R_1)$ are true. From the second rule of $\Pi^{\mathbf{A}}_{\varphi}$ we see that $G_{\psi}(s^L_1)$ and $G_{\psi}(s^R_1)$ hold. By the inductive hypothesis we get $\mathcal{K}, s^L_1 \models \psi$ and $\mathcal{K}, s^R_1 \models \psi$ (ψ is a state formula). Let $\pi = s_0, s_1, s_2, \ldots$ be an arbitrary path with initial state s_0. The fact that s_0 has both a left successor s^L_1 and a right successor s^R_1 implies that either s^L_1 or s^R_1 is the second state of every such path. Suppose that there exists a path $\pi = s_0, s_1, s_2, \ldots$ with initial state s_0 such that $\mathcal{K}, \pi \not\models \varphi$. If that were the case, then $\mathcal{K}, \pi^1 \not\models \psi$, where $\pi^1 = s_1, s_2, \ldots$. But $s_1 = s^L_1$ or $s_1 = s^R_1$, which means that $\mathcal{K}, s^L_1 \not\models \psi$ or $\mathcal{K}, s^R_1 \not\models \psi$, either of which contradicts the inductive hypothesis.
2. If $\varphi \equiv \psi_1 \mathbf{U} \psi_2$, where ψ_1 and ψ_2 are state formulas, then the corresponding corresponding existential and universal programs are these of Definition 6.2.(2).
   - The existential case:
     Suppose that $s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}})$. From the rules of the program $\Pi^{\mathbf{E}}_{\varphi}$ we see that there exists a s_i (possibly $s_i = s_0$) such that $G_{\psi_2}(s_i)$ holds. Further, there exists a sequence of states $s_0, s_1, \ldots, s_i$ such that for every r $(0 \leq r < i)$, $G_{\psi_1}(s_r)$ holds and at least one of $S_0(s_r, s_{r+1})$ or $S_1(s_r, s_{r+1})$ is true. By the inductive hypothesis we get $\mathcal{K}, s_i \models \psi_2$ and $\mathcal{K}, s_j \models \psi_1$ $(0 \leq j \leq i-1)$ (because ψ_1 and ψ_2 are state formulas). Let $\pi = s_0, s_1, s_2, \ldots, s_i, \ldots$ be any path with initial segment $s_0, s_1, \ldots, s_i$. Clearly, then $\mathcal{K}, \pi^i \models \psi_2$ and $\mathcal{K}, \pi^j \models \psi_1$ $(0 \leq j \leq i-1)$, i.e., $\mathcal{K}, \pi \models \varphi$.
   - The universal case:
     Let us assume now that $s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$. Let us define $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n)$ to be the set of ground facts for $G^{\mathbf{A}}_{\varphi}$ that have been computed in the first n rounds of the evaluation of program $\Pi^{\mathbf{A}}_{\varphi}$[3]. We shall prove that for every $t \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n)$, $\mathcal{K}, \pi \models \varphi$ for every path $\pi = t_0, t_1, t_2, \ldots$ with initial state $t_0 = t$. We use induction on the number of rounds n.
     (a) If $n = 1$, then t appears in $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$ due to the first rule of $\Pi^{\mathbf{A}}_{\varphi}$, i.e., $t \in G_{\psi_2}(D_{\mathcal{K}})$. By the inductive hypothesis with respect to ψ_2 we get that $\mathcal{K}, t \models \psi_2$ (recall that ψ_2 is a state formula), which trivially implies that $\mathcal{K}, \pi \models \varphi$ for every path $\pi = t_0, t_1, t_2, \ldots$ with initial state $t_0 = t$.
     (b) We show now that the claim holds for $n+1$, assuming that it holds for n. We examine only

[3]For a detailed treatment of the bottom-up evaluation of Datalog programs see [23].

the case where node t has exactly two successors t^L and t^R, since the case where t has only one successor is identical. Without loss of generality we may assume that t first appeared in $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n+1)$ during round $n+1$. This must have happened due to the third rule of $\Pi^{\mathbf{A}}_{\varphi}$: $G^{\mathbf{A}}_{\varphi}(x) \longleftarrow G_{\psi_1}(x),\ S_0(x,y),\ S_1(x,z),\ G^{\mathbf{A}}_{\varphi}(y),\ G^{\mathbf{A}}_{\varphi}(z)$. This implies that $t \in G_{\psi_1}(D_{\mathcal{K}})$ and both t^L and t^R belong to $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n)$. Hence, by invoking the inductive hypothesis with respect to the number of rounds, we get that $\mathcal{K}, t \models \psi_1$, $\mathcal{K}, \pi \models \varphi$ for every path $\pi^{1,L} = t_1^L, t_2^L, \ldots$ with initial state $t_1^L = t^L$ and $\mathcal{K}, \pi \models \varphi$ for every path $\pi^{1,R} = t_1^R, t_2^R, \ldots$ with initial state $t_1^R = t^R$. By combining all these, we conclude that $\mathcal{K}, \pi \models \varphi$ for every path $\pi = t_0, t_1, t_2, \ldots$ with initial state $t_0 = t$.

The bottom-up evaluation of Datalog programs guarantees that there exists $n \in \mathbb{N}$ such that $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n) = G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, r)$ for every $r > n$, i.e., $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}) = G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n)$.

3. If $\varphi \equiv \psi_1 \widetilde{\mathbf{U}} \psi_2$, where ψ_1 and ψ_2 are state formulas, then the corresponding existential and universal programs are shown in Definition 6.2.(3).

   - The existential case:
     Let us assume that $s_0 \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}})$. Let us define $G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}}, n)$ to be the set of ground facts for $G^{\mathbf{E}}_{\varphi}$ that have been computed in the first n rounds of the evaluation of program $\Pi^{\mathbf{E}}_{\varphi}$. We shall prove that for every $t \in G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}}, n)$, there exists a path $\pi = t_0, t_1, t_2, \ldots$ with initial state $t_0 = t$, such that $\mathcal{K}, \pi \models \varphi$. We use induction on the number of rounds n.
     (a) If $n = 1$, then t appears in $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$ due to either the first rule, i.e., $t \in G_{\psi_1}(D_{\mathcal{K}}) \cap G_{\psi_2}(D_{\mathcal{K}})$, or to the second rule, i.e., $t \in TC_{\psi_2}(D_{\mathcal{K}})$. In the first case, the inductive hypothesis pertaining to ψ_1 and ψ_2, implies that $\mathcal{K}, t \models \psi_1 \wedge \psi_2$, which immediately implies that $\mathcal{K}, \pi \models \varphi$ for any path $\pi = t_0, t_1, t_2, \ldots$ with initial state $t_0 = t$. In the second case, we use Lemma 6.4.(2), which asserts the existence of a finite sequence $t_0, t_1, \ldots, t_k$ of states, such that $t_0 = t_k = t$ and $\mathcal{K}, t_j \models \psi_2$, $0 \leq j \leq k$. Consider the path $\pi = (t_0, t_1, \ldots, t_k)^{\omega}$; for this path we have $\mathcal{K}, \pi \models \varphi$.
     (b) We show now that the claim holds for $n+1$, assuming that it holds for n. We focus on the case where node t has exactly two successors t^L and t^R (the case where t has only one successor is similar). We may further assume that t first appeared in $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n+1)$ during round $n+1$. This can only have occurred because of the third or fourth rule of $\Pi^{\mathbf{E}}_{\varphi}$. Then $t \in G_{\psi_2}(D_{\mathcal{K}})$ and at least one of t^L and t^R belongs to $G^{\mathbf{E}}_{\varphi}(D_{\mathcal{K}}, n)$. Without loss of generality, we assume that $t^L \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n)$. By the inductive hypothesis, we know that $\mathcal{K}, t^L \models \psi_2$ and that there exists a path $\pi^1 = t_1, t_2, \ldots$ with initial state $t_1 = t^L$, such that $\mathcal{K}, \pi^1 \models \varphi$. Immediately then we conclude that $\mathcal{K}, \pi \models \varphi$, for the path $\pi = t_0, t_1, t_2, \ldots$ with $t_0 = t$.
     The bottom-up evaluation of Datalog programs guarantees that there exists $n \in \mathbb{N}$ such that $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n) = G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, r)$ for every $r > n$, i.e., $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}) = G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, n)$.

   - The universal case:
     Let us suppose now that $s_0 \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$. Let us define $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, k)$ to be the set of ground facts for $G^{\mathbf{A}}_{\varphi}$ that have been computed after k rounds of the evaluation of program $\Pi^{\mathbf{A}}_{\varphi}$. We shall prove with simultaneous induction on the number of rounds k two things:
     (1) Let $s \in G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, k)$ and let $\pi = s_0, s_1, \ldots$ be an arbitrary path with initial state $s_0 = s$; then $\mathcal{K}, \pi \models \psi_1 \widetilde{\mathbf{U}} \psi_2$
     (2) Let $PL_{\psi_2}(t, k)$ hold (here of course $k \leq |W|$) and let $\varrho = t_0, t_1, \ldots, t_k, \ldots$ be an arbitrary path with initial state $t_0 = t$; then either $\mathcal{K}, \varrho \models \psi_1 \widetilde{\mathbf{U}} \psi_2$ or $\mathcal{K}, t_j \models \psi_2$, for $0 \leq j \leq k$.
     (a) If $k = 1$, then s appears in $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}})$ due to the first rule[4] of $\Pi^{\mathbf{A}}_{\varphi}$, i.e., $s \in G_{\psi_1}(D_{\mathcal{K}}) \cap G_{\psi_2}(D_{\mathcal{K}})$. Hence, $\mathcal{K}, \pi \models \psi_1 \widetilde{\mathbf{U}} \psi_2$, where $\pi = s_0, s_1, \ldots$ is any path with initial state $s_0 = s$. Similarly, $PL_{\psi_2}(t, 1)$ can only be derived by the ninth or tenth rule of $\Pi^{\mathbf{A}}_{\varphi}$. In any case, these rules imply that for every path $\pi = t_0, t_1, \ldots$ with initial state $t_0 = t$ we have that $\mathcal{K}, t_0 \models \psi_2$ and $\mathcal{K}, t_1 \models \psi_2$.
     (b) We show now that the claim holds for $k+1$, assuming that it holds for k. We consider the case where states s and t have exactly two successors s^L, s^R and t^L, t^R, respectively.
     (i) Initially, we shall consider the case where $k + 1 \leq |W|$.

     We may assume that s first appeared in $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, k+1)$ during round $k+1$; then s must have appeared because of the third rule $G^{\mathbf{A}}_{\varphi}(x) \longleftarrow G_{\psi_2}(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{\varphi}(y), G^{\mathbf{A}}_{\varphi}(z)$. Hence, $s \in G_{\psi_2}(D_{\mathcal{K}})$ and both of s^L and s^R belong to $G^{\mathbf{A}}_{\varphi}(D_{\mathcal{K}}, k)$. It is important to stress that in this case, s cannot arise from an application of the fourth rule of $\Pi^{\mathbf{A}}_{\varphi}$ because the first time this may happen is at round $|W|+1$. Hence, by the inductive hypothesis, we get that $\mathcal{K}, s \models \psi_2$, $\mathcal{K}, \pi^{1,L} \models \varphi$ for every path $\pi^{1,L} = s_1^L, s_2^L, \ldots$ with initial state $s_1^L = s^L$ and $\mathcal{K}, \pi^{1,R} \models \varphi$ for every path $\pi^{1,R} = s_1^R, s_2^R, \ldots$ with initial state $s_1^R = t^R \Rightarrow \mathcal{K}, \pi \models \varphi$ for every path $\pi = s_0, s_1, s_2, \ldots$ with initial state $s_0 = s$.

     Moreover, if $PL_{\psi_2}(t, k+1)$ holds, then one of the following must also hold:
     - $PL_{\psi_2}(t^L, k)$ and $PL_{\psi_2}(t^R, k)$,
     - $G^{\mathbf{A}}_{\varphi}(t^L)$ and $PL_{\psi_2}(t^R, k)$, or
     - $PL_{\psi_2}(t^L, k)$ and $G^{\mathbf{A}}_{\varphi}(t^R)$.

     Thus, by the inductive hypothesis, we know that:
     - $\mathcal{K}, t \models \psi_2$, and
     - for every path $\varrho^{1,L} = t_1^L, t_2^L, \ldots, t_{k+1}^L, \ldots$ with initial state $t_1^L = t^L$ either $\mathcal{K}, \varrho^{1,L} \models \varphi$ or $\mathcal{K}, t_j^L \models \psi_2$ $(1 \leq j \leq k+1)$, and
     - for every path $\varrho^{1,R} = t_1^R, t_2^R, \ldots, t_{k+1}^R, \ldots$ with

[4] We assume of course that $|W| > 1$ because the case where $|W| = 1$, that is the database contains only one element, is trivial.

initial state $t_1^R = t^R$ either $\mathcal{K}, \varrho^{1,R} \models \varphi$ or $\mathcal{K}, t_j^R \models \psi_2$ $(1 \leq j \leq k+1)$.

Taking all these into account, we conclude that for every path $\varrho = t_0, t_1, t_2, \ldots, t_{k+1}, \ldots$ with initial state $t_0 = t$ either $\mathcal{K}, \varrho \models \varphi$ or $\mathcal{K}, t_j \models \psi_2$ $(0 \leq j \leq k+1)$.
(ii) Finally, we examine the case where $k+1 > |W|$.

In this case, s may belong to $G_\varphi^{\mathbf{A}}(D_\mathcal{K}, k+1)$ either due to the third rule ($G_\varphi^{\mathbf{A}}(x) \longleftarrow G_{\psi_2}(x)$, $S_0(x,y)$, $S_1(x,z)$, $G_\varphi^{\mathbf{A}}(y)$, $G_\varphi^{\mathbf{A}}(z)$) or due to the fourth rule ($G_\varphi^{\mathbf{A}}(x) \longleftarrow PL_{\psi_2}(x, \mathbf{c}_{max})$). If it is due to the third rule, then $s \in G_{\psi_2}(D_\mathcal{K})$ and both of s^L and s^R belong to $G_\varphi^{\mathbf{A}}(D_\mathcal{K}, k)$, and the proof is the same as above. So we suppose that s is due to the fourth rule, i.e., $PL_{\psi_2}(s, |W|)$ is true. As we have already proved, this implies that given any path $\pi = s_0, s_1, s_2, \ldots, s_{|W|}, \ldots$ with initial state $s_0 = s$, either $\mathcal{K}, \pi \models \psi_1 \widetilde{\mathbf{U}} \psi_2$ or $\mathcal{K}, s_j \models \psi_2$, for $0 \leq j \leq |W|$. In order to arrive at a contradiction, let us assume that there exists a path $\varrho = s_0, s_1, s_2, \ldots$ with initial state $s_0 = s$ such that $\mathcal{K}, \varrho \not\models \psi_1 \widetilde{\mathbf{U}} \psi_2 \Rightarrow \mathcal{K}, \varrho \models \neg(\psi_1 \widetilde{\mathbf{U}} \psi_2) \Rightarrow \mathcal{K}, \varrho \models \neg\psi_1 \mathbf{U} \neg\psi_2 \Rightarrow \mathcal{K}, \varrho^j \models \neg\psi_2$, for $\varrho^j = s_j, s_{j+1}, s_{j+2}, \ldots$. We examine the case where $j > |W|$ (if $j \leq |W|$, then a contradiction is immediate). Consider the initial segment $s_0, s_1, s_2, \ldots, s_{|W|}, \ldots, s_j$ of ϱ; from Lemma 6.1 we get that in the aforementioned sequence there exists a state s' such that $s' = s_k = s_l$, $0 \leq k < l \leq j$. So we can get the sorter initial segment $s_0, \ldots, s_{k-1}, s_l, s_{l+1}, \ldots, s_j$. Now if the number of states in this new initial segment is less than or equal to $|W|$, we stop; otherwise we keep applying the same technique until we reduce sufficiently the length of the initial segment. So, we can eventually produce an initial segment $t_0, \ldots, t_m$, where $t_0 = s$, $t_m = s_j$ and $m \leq |W|$. But this contradicts the inductive hypothesis.

The bottom-up evaluation of Datalog programs guarantees that there exists $n \in \mathbb{N}$ such that $G_\varphi^{\mathbf{A}}(D_\mathcal{K}, n) = G_\varphi^{\mathbf{A}}(D_\mathcal{K}, r)$ for every $r > n$, i.e., $G_\varphi^{\mathbf{A}}(D_\mathcal{K}) = G_\varphi^{\mathbf{A}}(D_\mathcal{K}, n)$. ⊣

7. EXAMPLES

In this section, we give two more examples to illustrate the more complicated features of our translation.

EXAMPLE 7.1. *Let us consider a following formula that contains the modality $\widetilde{\mathbf{U}}$ in an existential context, such as $\varphi \equiv \mathbf{E}\bigcirc p \wedge \mathbf{E}(q\widetilde{\mathbf{U}}r)$. The Datalog program Π_φ that corresponds to φ can be constructed in four easy steps:*

1. *The programs that correspond to the state formulas p, q and r are:*
   - Π_p : $G_p(x) \longleftarrow P(x)$
   - Π_q : $G_q(x) \longleftarrow Q(x)$
   - Π_r : $G_r(x) \longleftarrow R(x)$
2. *The existential programs $\Pi^{\mathbf{E}}_{\bigcirc p}$ and $\Pi^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}$ are:*
   - $\Pi^{\mathbf{E}}_{\bigcirc p}$:
     $(1) : G^{\mathbf{E}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), G_p(y)$
     $(2) : G^{\mathbf{E}}_{\bigcirc p}(x) \longleftarrow S_1(x,y), G_p(y)$
     $(3) : G_p(x) \longleftarrow P(x)$
   - $\Pi^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}$:
     $(1) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_q(x), G_r(x)$
     $(2) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow TC_r(x,x)$
     $(3) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(y)$
     $(4) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_1(x,y), G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(y)$
     $(5) : TC_r(x,y) \longleftarrow G_r(x), S_0(x,y), G_r(x)$
     $(6) : TC_r(x,y) \longleftarrow G_r(x), S_1(x,y), G_r(x)$
     $(7) : TC_r(x,y) \longleftarrow G_r(x), S_0(x,y), TC_r(x)$
     $(8) : TC_r(x,y) \longleftarrow G_r(x), S_1(x,y), TC_r(x)$
     $(9) : G_q(x) \longleftarrow Q(x)$
     $(10) : G_r(x) \longleftarrow R(x)$
3. *The programs for the state formulas $\mathbf{E}\bigcirc p$ and $\mathbf{E}(q\widetilde{\mathbf{U}}r)$ are $\Pi_{\mathbf{E}\bigcirc p}$ and $\Pi_{\mathbf{E}(q\widetilde{\mathbf{U}}r)}$ respectively:*
   - $\Pi_{\mathbf{E}\bigcirc p}$:
     $(1) : G_{\mathbf{E}\bigcirc p}(x) \longleftarrow G^{\mathbf{E}}_{\bigcirc p}(x)$
     $(2) : G^{\mathbf{E}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), G_p(y)$
     $(3) : G^{\mathbf{E}}_{\bigcirc p}(x) \longleftarrow S_1(x,y), G_p(y)$
     $(4) : G_p(x) \longleftarrow P(x)$
   - $\Pi_{\mathbf{E}(q\widetilde{\mathbf{U}}r)}$:
     $(1) : G_{\mathbf{E}(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x)$
     $(2) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_q(x), G_r(x)$
     $(3) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow TC_r(x,x)$
     $(4) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(y)$
     $(5) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_1(x,y), G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(y)$
     $(6) : TC_r(x,y) \longleftarrow G_r(x), S_0(x,y), G_r(x)$
     $(7) : TC_r(x,y) \longleftarrow G_r(x), S_1(x,y), G_r(x)$
     $(8) : TC_r(x,y) \longleftarrow G_r(x), S_0(x,y), TC_r(x)$
     $(9) : TC_r(x,y) \longleftarrow G_r(x), S_1(x,y), TC_r(x)$
     $(10) : G_q(x) \longleftarrow Q(x)$
     $(11) : G_r(x) \longleftarrow R(x)$
4. *Finally, the program $\Pi_{\mathbf{E}\bigcirc p \wedge \mathbf{E}(q\widetilde{\mathbf{U}}r)}$ for the state formula $\mathbf{E}\bigcirc p \wedge \mathbf{E}(q\widetilde{\mathbf{U}}r)$ is:*
   - $\Pi_{\mathbf{E}\bigcirc p \wedge \mathbf{E}(q\widetilde{\mathbf{U}}r)}$:
     $(1) : G_{\mathbf{E}\bigcirc p \wedge \mathbf{E}(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_{\mathbf{E}\bigcirc p}(x)(x), G_{\mathbf{E}(q\widetilde{\mathbf{U}}r)}(x)$
     $(2) : G_{\mathbf{E}\bigcirc p}(x) \longleftarrow G^{\mathbf{E}}_{\bigcirc p}(x)$
     $(3) : G^{\mathbf{E}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), G_p(y)$
     $(4) : G^{\mathbf{E}}_{\bigcirc p}(x) \longleftarrow S_1(x,y), G_p(y)$
     $(5) : G_p(x) \longleftarrow P(x)$
     $(6) : G_{\mathbf{E}(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x)$
     $(7) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_q(x), G_r(x)$
     $(8) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow TC_r(x,x)$
     $(9) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(y)$
     $(10) : G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_1(x,y), G^{\mathbf{E}}_{(q\widetilde{\mathbf{U}}r)}(y)$
     $(11) : TC_r(x,y) \longleftarrow G_r(x), S_0(x,y), G_r(x)$
     $(12) : TC_r(x,y) \longleftarrow G_r(x), S_1(x,y), G_r(x)$
     $(13) : TC_r(x,y) \longleftarrow G_r(x), S_0(x,y), TC_r(x)$
     $(14) : TC_r(x,y) \longleftarrow G_r(x), S_1(x,y), TC_r(x)$
     $(15) : G_q(x) \longleftarrow Q(x)$
     $(16) : G_r(x) \longleftarrow R(x)$ ◀

EXAMPLE 7.2. *Let us consider a following formula that contains the modality $\widetilde{\mathbf{U}}$ in a universal context, such as $\varphi \equiv \mathbf{A}\bigcirc p \wedge \mathbf{A}(q\widetilde{\mathbf{U}}r)$. The Datalog program Π_φ that corresponds to φ can be constructed in four easy steps:*

1. *The programs that correspond to the state formulas p, q and r are the following:*

- $\Pi_p:\ G_p(x) \longleftarrow P(x)$
- $\Pi_q:\ G_q(x) \longleftarrow Q(x)$
- $\Pi_r:\ G_r(x) \longleftarrow R(x)$

2. *The universal programs $\Pi^{\mathbf{A}}_{\bigcirc p}$ and $\Pi^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}$ are:*

- $\left\{\begin{array}{l}(1): G^{\mathbf{A}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), \neg S_1(x,z), G_p(y)\\ (2): G^{\mathbf{A}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), S_1(x,z), G_p(y), G_p(z)\\ (3): G_p(x) \longleftarrow P(x)\end{array}\right.$

- $\left\{\begin{array}{l}(1): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_q(x), G_r(x)\\ (2): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y)\\ (3): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(z)\\ (4): G^{\mathbf{A}}_{q\widetilde{\mathbf{U}}r}(x) \longleftarrow PL_r(x, \mathbf{c}_{max})\\ (5): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), PL_r(y, n-1), n \leq \mathbf{c}_{max}\\ (6): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), PL_r(y,n-1), PL_r(z,n-1), n \leq \mathbf{c}_{max}\\ (7): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y), PL_r(z,n-1), n \leq \mathbf{c}_{max}\\ (8): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), PL_r(y,n-1), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(z), n \leq \mathbf{c}_{max}\\ (9): PL_r(x,1) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), G_r(y)\\ (10): PL_r(x,1) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G_r(y), G_r(z)\\ (11): G_q(x) \longleftarrow Q(x)\\ (12): G_r(x) \longleftarrow R(x)\end{array}\right.$

3. *The programs for the state formulas $\mathbf{A}\bigcirc p$ and $\mathbf{A}(q\widetilde{\mathbf{U}}r)$ are $\Pi_{\mathbf{A}\bigcirc p}$ and $\Pi_{\mathbf{A}(q\widetilde{\mathbf{U}}r)}$ respectively:*

- $\left\{\begin{array}{l}(1): G_{\mathbf{A}\bigcirc p}(x) \longleftarrow G^{\mathbf{A}}_{\bigcirc p}(x)\\ (2): G^{\mathbf{A}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), \neg S_1(x,z), G_p(y)\\ (3): G^{\mathbf{A}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), S_1(x,z), G_p(y), G_p(z)\\ (4): G_p(x) \longleftarrow P(x)\end{array}\right.$

- $\left\{\begin{array}{l}(1): G_{\mathbf{A}(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x)\\ (2): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_q(x), G_r(x)\\ (3): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y)\\ (4): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(z)\\ (5): G^{\mathbf{A}}_{q\widetilde{\mathbf{U}}r}(x) \longleftarrow PL_r(x, \mathbf{c}_{max})\\ (6): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), PL_r(y, n-1), n \leq \mathbf{c}_{max}\\ (7): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), PL_r(y,n-1), PL_r(z,n-1), n \leq \mathbf{c}_{max}\\ (8): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y), PL_r(z,n-1), n \leq \mathbf{c}_{max}\\ (9): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), PL_r(y,n-1), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(z), n \leq \mathbf{c}_{max}\\ (10): PL_r(x,1) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), G_r(y)\\ (11): PL_r(x,1) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G_r(y), G_r(z)\\ (12): G_q(x) \longleftarrow Q(x)\\ (13): G_r(x) \longleftarrow R(x)\end{array}\right.$

4. *Finally, the program $\Pi_{\mathbf{A}\bigcirc p \wedge \mathbf{A}(q\widetilde{\mathbf{U}}r)}$ for the state formula $\mathbf{A}\bigcirc p \wedge \mathbf{A}(q\widetilde{\mathbf{U}}r)$ is:*

- $\left\{\begin{array}{l}(1): G_{\mathbf{A}(\bigcirc p)\wedge\mathbf{A}(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_{\mathbf{A}(\bigcirc p)}(x), G_{\mathbf{A}(q\widetilde{\mathbf{U}}r)}(x)\\ (2): G_{\mathbf{A}\bigcirc p}(x) \longleftarrow G^{\mathbf{A}}_{\bigcirc p}(x)\\ (3): G^{\mathbf{A}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), \neg S_1(x,z), G_p(y)\\ (4): G^{\mathbf{A}}_{\bigcirc p}(x) \longleftarrow S_0(x,y), S_1(x,z), G_p(y), G_p(z)\\ (5): G_p(x) \longleftarrow P(x)\\ (6): G_{\mathbf{A}(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x)\\ (7): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_q(x), G_r(x)\\ (8): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y)\\ (9): G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(x) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(z)\\ (10): G^{\mathbf{A}}_{q\widetilde{\mathbf{U}}r}(x) \longleftarrow PL_r(x, \mathbf{c}_{max})\\ (11): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), PL_r(y, n-1), n \leq \mathbf{c}_{max}\\ (12): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), PL_r(y,n-1), PL_r(z,n-1), n \leq \mathbf{c}_{max}\\ (13): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(y), PL_r(z,n-1), n \leq \mathbf{c}_{max}\\ (14): PL_r(x,n) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), PL_r(y,n-1), G^{\mathbf{A}}_{(q\widetilde{\mathbf{U}}r)}(z), n \leq \mathbf{c}_{max}\\ (15): PL_r(x,1) \longleftarrow G_r(x), S_0(x,y), \neg S_1(x,z), G_r(y)\\ (16): PL_r(x,1) \longleftarrow G_r(x), S_0(x,y), S_1(x,z), G_r(y), G_r(z)\\ (17): G_q(x) \longleftarrow Q(x)\\ (18): G_r(x) \longleftarrow R(x)\end{array}\right.$ ◄

8. CONCLUSION AND FUTURE WORK

One of the most important problems for temporal logics is the computational complexity of model checking. It is known that, for CTL, model checking algorithms run in time $O(nm)$ ([4]), where n is the size of the model and m

the length of the formula. We do not try to make any statement on the computational complexity of our language however, as we believe that essentially we only use a fragment of $\text{Datalog}^{\neg}_{Succ}$ which can be syntactically characterized so that it has exactly the expressive power of CTL. We are currently working on this direction. Hence the computational complexity of query evaluation for this fragment is expected to be the same as the model checking complexity of CTL. However, it is an interesting exercise to prove directly the computational complexity of this fragment of $\text{Datalog}^{\neg}_{Succ}$.

In future work we plan to extend our approach to CTL*. CTL is a proper fragment of CTL* and is less expressive. CTL* was firstly defined ([6], [8]) as a means that would allow a more careful examination of the differences between branching and linear time temporal logic. More recently [18] has shown that CTL* has the same expressive power as the class of bisimulation invariant properties expressible in monadic path logic. In [15, 9] CTL* is translated in the modal μ-calculus and in [11] CTL* is translated in the fragment of the transitive closure logic FO(TC), with two variables.

The logic CTL* combines both branching-time and linear time operators ([8]). A path quantifier **E** (for some path) or **A** (for all paths) can prefix an assertion built from the set of atomic proposition using the usual Boolean connectives and the temporal operators $\bigcirc$ (next time), **U** (until) and $\widetilde{\mathbf{U}}$ (duality of until). There are two types of formulas in CTL*. The state formulas whose satisfaction is related to a specific state and path formulas whose satisfaction is related to a specific path. Therefore, CTL* state formulas are defined as follows:

- All atomic propositions, $\top$ and $\bot$ are CTL* state formulas,
- If φ and ψ are CTL* state formulas, then $\neg\varphi, (\varphi \wedge \psi), (\varphi \vee \psi)$ are also CTL* state formulas, and
- If φ is a **path** formula, then $\mathbf{A}\varphi$ and $\mathbf{E}\varphi$ are CTL* state formulas.

The CTL* **path** formulas are defined to be:

- A **state** formula is a path formula,
- If φ and ψ are CTL* path formulas, then $\neg\varphi, (\varphi \wedge \psi), (\varphi \vee \psi)$ are also CTL* path formulas, and
- If φ and ψ are path formulas, then $\bigcirc\varphi$, $(\varphi\mathbf{U}\psi)$ and $(\varphi\widetilde{\mathbf{U}}\psi)$ are CTL* path formulas.

The logic CTL* consists of the set of state formulas generated by the above rules. The logic CTL is a restricted subset of CTL*. In CTL the temporal operators $\bigcirc$, **U** and $\widetilde{\mathbf{U}}$ must be immediately preceded by a path quantifier. Formally, it is the subset of CTL* obtained by restricting the path formulas to be $\bigcirc\psi$, $\psi\mathbf{U}\varphi$, or $\psi\widetilde{\mathbf{U}}\varphi$ where ψ and φ are CTL state formulas.

Acknowledgements
We thank Scott Smolka for providing useful input and comments. We also thank Alex Shvartsman for helping out with the format.

9. REFERENCES

[1] F. Afrati, T. Andronikos, E. Foustoucos, and I. Guessarian. Ctl et al. vs. monadic inf-datalog. In *Int. Workshop on Logic and Complexity in Computer Science LCCS 2001, France*, pages 17–36, 2001.

[2] F. Bancilhon and R. Ramakrishnan. An amateur's introduction to recursive query processing strategies. In *Proc. ACM Conf. on Management of Data, Washington*, pages 16–52.

[3] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer-Verlag, 1990.

[4] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. In *ACM TOPLAS,8*, pages 244–263, 1986.

[5] B. Cui, Y. Dong, X. Du, K. Kumar, C. Ramakrishnan, I. Ramakrishnan, A. Roychoudhury, S. A. Smolka, and D. Warren. Logic programming and model checking. In *PLAP/ALP'98*, volume 1490, pages 1–20. LNCS, Springer.

[6] E.A.Emerson and A.P.Sistla. Deciding full branching time logic. *Information and Control*, 61(3):175–201, 1984.

[7] E. A. Emerson. *Temporal and modal logic*. Handbook of Theoretical Computer Science, MIT Press, 1990.

[8] E. A. Emerson and J. Halpern. Sometimes and not never revisited: On branching versus linear time. 33(1):151–178, 1986.

[9] E. A. Emerson and C. Lei. Efficient model checking in fragments of the propositional μ-calculus. In *In Proc. of 1rst Symposium on Logic in Computer Science*, pages 267–278, 1986.

[10] G. Gottlob, E. Grädel, and H. Veith. Datalog LITE: Temporal versus deductive reasoning in verification. In J. Minker, editor, *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14–16, 1999*, College Park, Maryland, 1999. Computer Science Department, University of Maryland. citeseer.nj.nec.com/gottlob98datalog.html.

[11] N. Immermann and M. Vardi. Model checking and transitive-closure logic. In *CAV'97*, volume 1254, pages 291–302. LNCS, Springer, 1997. citeseer.nj.nec.com/immerman97model.html.

[12] P. Kanellakis and S. Smolka. Ccs expressions, finite state processes, and three problems of equivalence. In *Proceedings, 2nd Annual ACM Symposium on Principles of Distributed Computing, Montreal, Canada*, pages 228–240, 1983.

[13] P. Kanellakis and S. Smolka. On the analysis of cooperation and antagonism in networks of communicating processes. *Algorithmica 3*, pages 421–450, 1988.

[14] P. Kanellakis and S. Smolka. Ccs expressions, finite state processes and three problems of equivalence. *Information and Computation*, 86:43–68, 1990.

[15] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[16] O. Kupferman, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000. citeseer.nj.nec.com/kupferman98automatatheoretic.html.

[17] L. Lamport. Sometimes is sometimes "not never"-on

the temporal logic of programs. In *In Proc. 7th ACM Symposium on Principles of programming languages*, pages 174–185, 1980.

[18] F. Moller and A. Rabinovich. On the expressive power of ctl^*. In *In Proc. LICS'99, Trento, Italy*, 1999.

[19] A. Pnueli. The temporal logic of programs. In *In Proc. 18th IEEE Symposium on Foundation of Computer Science*, pages 46–57, 1977.

[20] A. Podelski, W. Charatonik, and M. Muller. Set-based analysis of reactive infinite-state systems. In *TACAS'98*. LNCS, Springer, 1998. citeseer.nj.nec.com/58196.html.

[21] Y. S. Ramakrishna, C. R. Ramakrishnan, I. V. Ramakrishnan, S. A. Smolka, T. Swift, and D. S. Warren. Efficient model checking using tabled resolution. In *Computer Aided Verification (CAV'97)*, volume 1254, pages 143–154. LNCS, Springer, 1997. citeseer.nj.nec.com/ramakrishna97efficient.html.

[22] N. Rescher and A. Urquhart. Temporal logic. *Springer-Verlag, New York*, 1971.

[23] J. Ullman. *Database and Knowledge-Base Systems, Volumes I and II*. Computer Science Press, 1989.

[24] M. Vardi. *An Automata-Theoretic approach to linear temporal logic*. Logics for Concurrency: Structure versus Automata, volume 1043 of Lecture Notes in Computer Science, pp. 238-266, Springer-Verlag, Berlin, 1996.

[25] S. Zhang, S. Smolka, and O. Sokolsky. On the parallel complexity of model checking in the modal mu-calculus. In *Proceedings of Ninth Annual IEEE Symposium on Logic in Computer Science, London*, pages 154–163, 1994.