Technische Universität Dresden

Department of Computer Science
Institute of Theoretical Computer Science
Chair of Automata Theory
Prof. Dr.-Ing. Franz Baader

A thesis in fulfillment of the requirements for the degree of

Bachelor of Science

Experimental Evaluation of a Bounded History Encoding

Thure Nebendahl

Mat.-No.: 4684126
Born September 12, 1999 in Bonn

First Reviewer: Dr.-Ing Stefan Borgwardt
Second Reviewer: Prof. Dr. Markus Krötzsch

Dresden, August 8, 2020

Declaration of Authorship

I hereby declare that I wrote the bachelor thesis I submitted today to the examination board of the Faculty of Computer Science on the topic:

Experimental Evaluation of a Bounded History Encoding

completely on my own and that I did not use any sources and tools other than those indicated. All thoughts taken directly or indirectly from external sources are properly denoted as such.

Dresden, August 8, 2020

Thure Nebendahl

Thure Nebendahl

Abstract

The reasoning task of a temporal version of *ontology-based data access* can be solved with the help of a *bounded history encoding* (BHE). By rewriting temporal queries, the reasoning task can be reduced to query answering over temporal databases. The BHE under consideration in this work is one approach to solve this task [1].

In this thesis, an implementation of this BHE is evaluated, to investigate its helpfulness in a practical application. Criteria to assess the degree of helpfulness are the size of the encoding, the time it takes to answer one query per time point, and the usefulness of the answers. The BHE is applied to observe an online automotive marketplace and then to answer a set of specified queries at each time point. The queries were explicitly chosen to help evaluate the BHE's degree of helpfulness. To provide useful answers, the BHE had to be extended by appropriate operators. The evaluation of the extended BHE shows that it can provide useful answers in a reasonable time. An upper bound was found for the size of the encoding, which determines from what number of time points on the BHE is superior.

Contents

List of Tables

List of Figures

Acronyms

BHE	Bounded History Encoding
OBDA	Ontology-Based Data Access
TQ	Temporal Query
PTQ	Practical Temporal Query
RTQ	Random Temporal Query
MTO	Metric Temporal Operator
SVTT	System-Versioned Temporal Tables

Introduction

This thesis deals with a *bounded history encoding* (BHE) as in [2] and its experimental application on an online automotive marketplace.

The theoretical framework for this thesis is set by [1]. In [1], a temporal version of *ontology-based data access* (OBDA) is considered. A generic temporal query language is presented. The reasoning task of the temporal OBDA is reduced to query answering over temporal databases, the so-called *temporal database monitoring problem.* Three approaches to solve this problem are presented, including an algorithm that constitutes a BHE. This work builds upon the latter approach. The algorithm will also be referred to as the BHE, since it is the only BHE considered here.

This thesis ties in with [1] and checks the applicability of the algorithm through a concrete application on publicly available data, collected from an online automotive marketplace. The investigation carried out is structured in four steps. Chapter 1 shows how the publicly accessible data is regularly extracted from an online portal into a database. In Chapter 2 an attempt is made to formulate practical and meaningful queries in the language of [1]. It is to be checked, whether the query language should be extended by additional operators. Supplementary to practical and meaningful queries, random queries are formulated based on the idea from [3]. This ensures that the results are valid for the entire language from [1]. The results of the second chapter are used in the third chapter to construct concrete extensions of the already existent query language from [1]. To analyze the algorithm, in Chapter 4 the extended language is applied to the extracted data from the online automotive marketplace. The goal of this chapter is to evaluate how well the queries can be answered with the help of the BHE [1]. Criteria for this are the size of the encoding, the time it takes to answer one query per time point, and the usefulness of the answers. Based on the above-mentioned investigations, the Conclusion summarizes the results to answer the research question, i.e. to which extent the BHE is helpful in a practical application.

Chapter 1

Discrete Data Stream

This chapter discusses in detail the *discrete data stream* used as a basis for the investigation of the algorithm from [1]. It explains where this data comes from, how it is retrieved, and how it can be processed. Finally, it explains how the already existent implementation of the BHE [4] can query the data.

The publicly accessible data, which is used as a discrete data stream, originates from the online automotive marketplace *AutoScout24* [5]. It provides data on many models of various brands. The selection of collected brands and models must be narrowed down so that the data remains manageable but still delivers noticeable results. It is assumed that it is irrelevant which exact brands and models are collected. Later, this thesis shows that this is true. A complete list of all models can be found in Appendix A. To create a real discrete data stream, the data is collected weekly for 10 weeks. This is the longest possible period for this thesis. To verify whether this affects the research result, the result of the BHE [1] that is applied to weekly collected data is compared to the result of the BHE [1], when applied to daily collected data. Collecting the data daily artificially increases the number of time points.

The data is collected by a scraper [6] implemented in Python, similar to the one presented in [7]. The scraper uses the modules *Beautiful Soup 4* [8], *pandas* [9], and *SQLalchemy* [10].

First, it is specified which aspects of an individual car the scraper [6] should cover. This is necessary because most of the data collected from a website does not contain any information about the car. Again, it is assumed that it is irrelevant which exact data is collected about the car. The most noteworthy attributes, since they are used in examples in this thesis, are *url*, *price*, and *deleted*. A complete list of all aspects can be found in Appendix A.

Second, the scraper [6] searches AutoScout24 [5] for a specific brand and model and saves the link to each offer the search yields. This is performed with the Beautiful Soup 4 *Soupstrainer* [8]. To ensure a consistent order of the offers, they are sorted by age, starting with the newest.

Third, once all links are saved, the scraper [6] searches every offer individually and extracts all the previously determined aspects from the HTML, again using Beautiful Soup 4 [8].

Fourth, the collected data is stored in a pandas *DataFrame* [9] to be loaded further into an *SQLite* database with SQLalchemy [10]. The corresponding table is called *autos*. This final step of storing the data in an SQLite database is required because the implementation of the BHE [4] operates on an SQLite database as well. This enables cross-language interaction between the scraper [6] and the implementation [4].

On AutoScout24 [5], the search results consist of a maximum of 20 pages, which then themselves contain a maximum of 20 offers. This maximum of 400 offers per search should not have a major impact on our research outcome. The number of brands still creates a data set with over 20,000 offers per week, which is expected to be sufficiently large to deliver noticeable results. Thus, the maximum of 400 offers helps to keep the data manageable.

However, it also presents some challenges. With some models, offers are added with a high frequency, so the 400 offers found since last week may all be new. To ensure the scraper [6] searches every offer from last week again, it scans through all previously found offers before searching for new ones. At the same time, reviewing each offer provides a better reference to check which offers have been deleted. If the scraper [6] tries to search for a deleted offer, an error message is returned. This provides more certainty than just assuming an offer has been deleted if the search results no longer contain that offer. The latter could be a consequence of frequently added new offers. To simplify matters, it is assumed that every deleted offer has been sold.

Although the BHE [1] only requires the data of the current time point, instead of replacing the database, we create a new database each time. Storing all the data brings many advantages for later analysis, such as

- by querying the data one query at a time, it is easier to analyze the impact of each query,
- each step can be repeated to check whether the results are reproducible, and
- it is possible to query any partial period.

Furthermore, the storage of all data allows a comparison of the BHE [1] with other established approaches.

Chapter 2

Queries

The previous chapter covered in detail how the data stream is processed so that queries can be executed on the data. This chapter now takes a closer look at the queries used for the investigation of the BHE [1]. It further discusses whether the queries specified here can be expressed in the language of [1].

The language of [1] consists of *temporal queries* (TQs). They can be based on any atemporal query language $\mathcal{Q}$.

Definition 2.0.1 (temporal query ct. Definition 3.2 in [1])**.** Given a query language $\mathcal{Q}$, *temporal $\mathcal{Q}$-queries* are built from $\mathcal{Q}$-queries as follows:

- every $\mathcal{Q}$-query ψ is a temporal $\mathcal{Q}$-query; and
- if ϕ_1 and ϕ_2 are temporal $\mathcal{Q}$-queries, then so are:
  - $\phi_1 \land \phi_2$ (conjunction), $\phi_1 \lor \phi_2$ (disjunction),
  - $\bigcirc\phi_1$ (strong next), $\bullet\phi_1$ (weak next),
  - $\bigcirc^-\phi_1$ (strong previous), $\bullet^-\phi_1$ (weak previous),
  - $\Box\phi_1$ (always), $\Box^-\phi_1$ (always in the past),
  - $\Diamond\phi_1$ (eventually), $\Diamond^-\phi_1$ (some time in the past),
  - $\phi_1\mathsf{U}\phi_2$ (until), and $\phi_1\mathsf{S}\phi_2$ (since).

The symbols $\bigcirc^-, \bullet^-, \Box^-, \Diamond^-$, and S are called *past operators*, the symbols $\bigcirc, \bullet, \Box, \Diamond$, and U are *future operators*.

Since the implementation of the BHE [4] uses an SQLite database, the atemporal query language $\mathcal{Q}$ in this thesis is SQL.

In this thesis, two different types of TQs are distinguished. First, meaningful *practical* TQs (PTQs) and second, *random* TQs (RTQs) based on the idea presented in [3].

2.1 Practical Temporal Queries

PTQs show whether the BHE [1] can give relevant answers to queries. This is necessary to find out if it has practical relevance and can be used, e.g. for market observation as it is done in this thesis. For reasons of readability, only the following selected queries are discussed as examples:

"Which cars cost less than 10,000 at the last time point and cost more than 10,000 at this time point?" (2.1)

"Which brands (or models) have an increased average price compared to the last time point?" (2.2)

"Which cars cost more than 1.000.000 at one time point during the last 6 time points?" (2.3)

A complete list of PTQs can be found in Appendix B.

2.2 Random Temporal Queries

RTQs are used to verify the impact of the queries on the overall research outcome. They are necessary because PTQs might not be sufficiently diverse to provide reliable results. RTQs can be constructed in any size.

Definition 2.2.1 (random temporal query ct. [3])**.** Let $n \in \mathbb{N}\backslash\{0\}$. Let Ops be a set of available operators. A *random* TQ of size n can be constructed by recursively constructing its subqueries as follows:

```
randomTQ($n$)
if $n = 0$ then
  return $\mathcal{Q}$-query $\psi$
else
  select a random operator $op$
  if $op$ is unary then
    return $op$(randomTQ($n - 1$))
  else // $op$ is binary
    select a random $m \in \mathbb{N} \setminus \{0\}, m < n$
    return (randomTQ($m$) $op$ randomTQ($n - m - 1$))
```

In this thesis, 100 RTQs of sizes 1 to 10 are considered. The individual RTQs are referenced by numbers. A complete list of generated RTQs can be found in Appendix B.

2.3 Expressibility

For RTQs, since only a predefined set of operators is used in the construction, any RTQ can be expressed in any desired language, i.e. the language of [1], if Ops is exactly the set of operators available in this language. In the following $\mathcal{Q}$-query ψ is defined as the SQL statement "SELECT *url* FROM *autos* WHERE NOT *deleted*" which returns every car that is not sold, just to be able to recognize the difference between a query result and the whole table.

The expressibility of PTQs is not as easy to assess. As mentioned in Chapter 1, *autos* is the table in which all offers are stored. The attribute *url* is distinct for each dataset and serves as an identifier for a car. Hence, if the query asks for cars, it will return a set of urls. This said, Query 2.1 can be translated into the language of [1] as follows:

- "[. . .] and [. . .]" indicates the conjunction operator from [1],
- "[. . .] at the last time point [. . .]" indicates either a strong previous or a weak previous operator from [1],
- "Which cars cost less than 10,000 [. . .]" indicates the SQL statement "SELECT *url* FROM *autos* WHERE *price* $<$ 10000", and
- "[. . .] more than 10,000 [. . .]" indicates the SQL statement "SELECT *url* FROM *autos* WHERE *price* $>$ 10000".

Thus, a resulting query in the language of [1] is

$$\text{“}\bigcirc^{-}(\text{SELECT } \mathit{url} \text{ FROM } \mathit{autos} \text{ WHERE } \mathit{price} < 10000)$$
$$\wedge \text{ SELECT } \mathit{url} \text{ FROM } \mathit{autos} \text{ WHERE } \mathit{price} > 10000\text{”}.$$

Query 2.2 can be rewritten as follows:

- "[. . .] compared to the last time point?" indicates the use of the conjunction operator from [1] with either a strong previous or a weak previous operator from [1] as one argument, and

- since the query needs to compare the value of one attribute at two different points in time it needs a restriction like "[...] WHERE *price*1 < *price*2" with *price*1 being the average price of the first time point and *price*2 the average price of the second time point.

Thus, a resulting query might be

"$\bigcirc^-$(SELECT *marke*, AVG(*price*) AS *price*1 FROM *autos*)
$\wedge$ SELECT *marke*, AVG(*price*) AS *price*2 FROM *autos* WHERE *price*1 < *price*2".

Notice, that this kind of more sophisticated filtering is not considered in [1] and therefore is also not considered in the implementation [4]. If expressed with the available operators in the language of [1], the implementation of the BHE [4] rewrites the query into the SQL statement

"SELECT * FROM *result_table_XY* NATURAL JOIN
(SELECT *marke*, AVG(*price*) AS *price*2 FROM *autos* WHERE *price*1 < *price*2)"

where *result_table_XY* denotes the table that contains the answer to the subquery

"$\bigcirc^-$(SELECT *marke*, AVG(*price*) AS *price*1 FROM *autos*)".

This statement is not a valid SQL statement, as there is no such column *price*1 in the second part of the statement. A corresponding valid SQL statement is

"SELECT * FROM (SELECT * FROM *result_table_XY* NATURAL JOIN
(SELECT *marke*, AVG(*price*) AS *price*2 FROM *autos*) WHERE *price*1 < *price*2)"

which shows, that more sophisticated filtering can be achieved by applying another SQL statement, that contains the wanted filter, to the result of the underlying query. Thus, to express this query in the language of [1], the language needs to be extended by a *filter operator*.

Furthermore, Query 2.3 can be rewritten as follows:

"SELECT *url* FROM *autos* WHERE *price* > 1000000
$\vee\bigcirc^-$(SELECT *url* FROM *autos* WHERE *price* > 1000000)
$\vee\bigcirc^-\bigcirc^-$(SELECT *url* FROM *autos* WHERE *price* > 1000000)
$\vee\bigcirc^-\bigcirc^-\bigcirc^-$(SELECT *url* FROM *autos* WHERE *price* > 1000000)
$\vee\bigcirc^-\bigcirc^-\bigcirc^-\bigcirc^-$(SELECT *url* FROM *autos* WHERE *price* > 1000000)
$\vee\bigcirc^-\bigcirc^-\bigcirc^-\bigcirc^-\bigcirc^-$(SELECT *url* FROM *autos* WHERE *price* > 1000000)"

This kind of query is highly complex when expressed in the language of [1]. To simplify matters, this query would need a time limit in the form of *metric temporal operators* (MTOs).

Lastly, the operators $\Box\phi$ (always), $\Box^{-}\phi$ (always in the past), $\Diamond\phi$ (eventually), $\Diamond^{-}\phi$ (sometime in the past) were introduced in [1] but not considered when defining the algorithm.

Consequently, it is necessary to extend the language of [1] and the implementation of the BHE [4] by these missing operators, a filter operator and MTOs.

Chapter 3

Extensions

It became clear that both the language from [1] and consequently the implementation of the BHE [4] must be extended to provide relevant answers to the PTQs specified in Chapter 2. In this chapter, the extensions are introduced one by one and examined for feasibility. The definitions of $\mathsf{eval}^n(\alpha)$, $\Phi_0(\psi)$ and $\Phi_i(\psi)$ from [1] are modified, upon which the modifications are proven to be correct.

3.1 Operators $\Box$, $\Box^-$, $\Diamond$, $\Diamond^-$

First, the algorithm presented in [1] is extended by the operators $\Box\phi$ (always), $\Box^-\phi$ (always in the past), $\Diamond\phi$ (eventually), $\Diamond^-\phi$ (sometime in the past). As a reminder, the semantics of these four TQs are defined as follows:

Definition 3.1.1 (semantics of temporal queries ct. Definition 3.3 in [1])**.** Let ϕ be a TQ, $\mathfrak{I} = (I_i)_{0\leq i\leq n}$ a sequence of interpretations over a common domain, $\mathfrak{a} : \mathsf{FVar}(\phi) \to \mathsf{N_C}$ a variable assignment, and i be an integer with $0 \leq i \leq n$. The *satisfaction relation* $\mathfrak{I}, i \models \mathfrak{a}(\phi)$ is defined by induction on the structure of ϕ as follows:

ϕ	$\mathfrak{I}, i \models \mathfrak{a}(\phi)$ iff
$\Box\phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all $k, i \leq k \leq n$
$\Box^-\phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all $k, 0 \leq k \leq i$
$\Diamond\phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some $k, i \leq k \leq n$
$\Diamond^-\phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some $k, 0 \leq k \leq i$

Table 3.1: Satisfaction relation of temporal queries $\Box, \Box^-, \Diamond$ and $\Diamond^-$

$\mathsf{FVar}(\phi)$ denotes the set of *free variables* of a TQ and is defined as the union of the sets $\mathsf{FVar}(\psi)$ of all queries ψ occurring in ϕ. $\mathsf{N_C}$ denotes a set of *constants*. If $\mathfrak{I}, i \models \mathfrak{a}(\phi)$, then $\mathfrak{a}$ is called an *answer* to ϕ w.r.t. $\mathfrak{I}$ at time point i. The set of all answers to ϕ w.r.t $\mathfrak{I}$ at time point i is denoted by $\mathsf{Ans}(\phi, \mathfrak{I}, i)$.

As in [1], it can be shown that

- $\Box\phi_1$ is equivalent to $\phi_1 \wedge \CIRCLE\Box\phi_1$, and
- $\Diamond\phi_1$ is equivalent to $\phi_1 \vee \Circle\Diamond\phi_1$.

Because of the way $\Box$ is defined in [1], $\CIRCLE$ has to be used instead of $\Circle$ with the only difference that $\CIRCLE$ is tautological at the last time point. It can similarly be shown that

- $\Box^-\phi_1$ is equivalent to $\phi_1 \wedge \CIRCLE^-\Box^-\phi_1$, and
- $\Diamond^-\phi_1$ is equivalent to $\phi_1 \vee \Circle^-\Diamond^-\phi_1$.

Analogously to the reason as mentioned above, $\CIRCLE^-$ has to be used instead of $\Circle^-$ with the only difference that $\CIRCLE^-$ is tautological at the first time point. Thus, at the last time point

- $\Box\phi_1$ is equivalent to ϕ_1 because $\CIRCLE\Box\phi_1$ is tautological, and
- $\Diamond\phi_1$ is equivalent to ϕ_1 because $\Circle\Diamond\phi_1$ does not have any answers,

and at the first time point

- $\Box^-\phi_1$ is equivalent to ϕ_1 because $\CIRCLE^-\Box^-\phi_1$ is tautological, and
- $\Diamond^-\phi_1$ is equivalent to ϕ_1 because $\Circle^-\Diamond^-\phi_1$ does not have any answers.

Proposition 3.1.2 (ct. Proposition 3.4 in [1]). *For $\mathfrak{a} : \mathsf{FVar}(\phi) \to \mathsf{N_C}$ and $0 \leq i \leq n$,*

1. $\mathfrak{I}, i \models \mathfrak{a}(\Box\phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *and*
   - $i < n$ *implies* $\mathfrak{I}, i+1 \models \mathfrak{a}(\Box\phi_1)$
2. $\mathfrak{I}, i \models \mathfrak{a}(\Box^-\phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *and*
   - $i > 0$ *implies* $\mathfrak{I}, i-1 \models \mathfrak{a}(\Box^-\phi_1)$

3. $\mathfrak{I}, i \models \mathfrak{a}(\Diamond\phi_1)$ *iff*
    - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or*
    - $i < n$ *and* $\mathfrak{I}, i+1 \models \mathfrak{a}(\Diamond\phi_1)$
4. $\mathfrak{I}, i \models \mathfrak{a}(\Diamond^{-}\phi_1)$ *iff*
    - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or*
    - $i > 0$ *and* $\mathfrak{I}, i-1 \models \mathfrak{a}(\Diamond^{-}\phi_1)$

Proof. To prove the above proposition, two equivalences are demonstrated here. The other two cases work similarly and can be found in Appendix C. The proof works mainly based on semantics.

1. $\Box\phi_1 \equiv \phi_1 \wedge \bullet\Box\phi_1$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\Box\phi_1) && (3.1)\\
\Leftrightarrow &\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, i \leq k \leq n && (3.2)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i < n \text{ implies} && (3.3)\\
&\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, i+1 \leq k \leq n) &&\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i < n \text{ implies } \mathfrak{I}, i+1 \models \mathfrak{a}(\Box\phi_1)) && (3.4)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \wedge \bullet\Box\phi_1) && (3.5)
\end{aligned}$$

(3.3) is equivalent to (3.2) because

- in case $i < n$, the query needs to be satisfied now, at time point i, and at all future time points k, $i+1 \leq k \leq n$, to be satisfied. Since $i < n$ is true, the satisfaction of future time points depends solely on the second part of the "implies"-statement; and
- in case $i = n$, the query needs to be satisfied now, at time point i, to be satisfied. There are no future time points k, $n+1 \leq k \leq n$. Since $i = n$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $n \leq k \leq n$, and $i < n$ is not true, thus the "implies"-statement does not affect satisfaction.

4. $\Diamond^{-}\phi_1 \equiv \phi_1 \vee \bigcirc^{-}\Diamond^{-}\phi_1$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\Diamond^{-}\phi_1) && (3.6)\\
\Leftrightarrow &\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, 0 \leq k \leq i && (3.7)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i > 0 \text{ and} && (3.8)\\
&\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, 0 \leq k \leq i-1)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i > 0 \text{ and } \mathfrak{I}, i-1 \models \mathfrak{a}(\Diamond^{-}\phi_1)) && (3.9)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \vee \bigcirc^{-}\Diamond^{-}\phi_1) && (3.10)
\end{aligned}$$

(3.8) is equivalent to (3.7) because

- in case $i > 0$, the query needs to be satisfied now, at time point i, or at any past time point k, $0 \leq k \leq i-1$, to be satisfied. Since $i > 0$ is true, the satisfaction of past time points depends solely on the second part of the "and"-statement; and
- in case $i = 0$, the query needs to be satisfied now, at time point i, to be satisfied. There are no past time points k, $0 \leq k \leq 0-1$. Since $i = 0$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $0 \leq k \leq 0$, and $i > 0$ is not true, thus the "and"-statement does not affect satisfaction.

$\square$

The semantics of the four operators can now be used to extend the algorithm specified in [1]. For this, the notation of *answer terms* is needed. Using the same simplification as in [1], the following assumes that $\mathsf{N_V}$, the set of *variables*, is finite and that the answers are of the form $\mathfrak{a} : \mathsf{N_V} \to \Delta$ instead of $\mathfrak{a} : \mathsf{FVar}(\phi) \to \Delta$. $\mathsf{Ans}(\phi, \mathfrak{I}^{(n)})$ refers to a set of mappings $\mathfrak{a} : \mathsf{N_V} \to \Delta$, i.e, a subset of $\Delta^{\mathsf{N_V}}$.

Definition 3.1.3 (answer term cf. Definition 6.1 in [1]). Let $\mathsf{FSub}(\phi)$ denote the set of all subqueries of ϕ of the form $\bigcirc\psi_1, \bullet\psi_1, \Box\psi_1, \Diamond\psi_1$ or $\psi_1 \mathsf{U} \psi_2$. For $j \geq 0$, we denote by Var_j^{ϕ} the set of all variables of the form x_j^{ψ} for $\psi \in \mathsf{FSub}(\phi)$. The set AT_{ϕ}^{i} of all *answer terms* for ϕ at $i \geq 0$ is the smallest set satisfying the following conditions:

- every set $A \subseteq \Delta^{\mathsf{N_V}}$ is an answer term for ϕ at i,
- every variable $x_j^{\psi} \in \mathsf{Var}_j^{\phi}$ with $j \leq i$ is an answer term for ϕ at i, and
- if α_1 and α_2 are answer terms for ϕ at i, then so are $\alpha_1 \cap \alpha_2$ and $\alpha_1 \cup \alpha_2$.

The functions $\mathsf{eval}^n : \mathsf{AT}^n_\phi \to 2^{\Delta^{\mathsf{N_V}}}, n \geq 0$ in [1] have then to be extended as follows:

α	$\mathsf{eval}^n(\alpha)$
$[\ldots]$	$[\ldots]$
$x_j^{\Box\psi_1}$ with $j < n$	$\mathsf{Ans}(\Box\psi_1, \mathfrak{I}^{(n)}, j+1)$
$x_j^{\Diamond\psi_1}$ with $j < n$	$\mathsf{Ans}(\Diamond\psi_1, \mathfrak{I}^{(n)}, j+1)$
$x_n^{\Box\psi_1}$	$\Delta^{\mathsf{N_V}}$
$x_n^{\Diamond\psi_1}$	$\emptyset$

Table 3.2: $\mathsf{eval}^n(\alpha)$ with $\Box, \Box^-, \Diamond$ and $\Diamond^-$

The function $\Phi_0(\psi) : \mathsf{Sub}(\phi) \to \mathsf{AT}^0_\phi$ in [1] has to be extended as follows:

ψ	$\Phi_0(\psi)$
$[\ldots]$	$[\ldots]$
$\Box\psi_1$	$\Phi_0(\psi_1) \cap x_0^{\Box\psi_1}$
$\Box^-\psi_1$	$\Phi_0(\psi_1)$
$\Diamond\psi_1$	$\Phi_0(\psi_1) \cup x_0^{\Diamond\psi_1}$
$\Diamond^-\psi_1$	$\Phi_0(\psi_1)$

Table 3.3: $\Phi_0(\psi)$ with $\Box, \Box^-, \Diamond$ and $\Diamond^-$

The function $\Phi_i^0(\psi) : \mathsf{Sub}(\phi) \to \mathsf{AT}^i_\phi,\ i > 0$ in [1] has to be extended as follows:

ψ	$\Phi_i^0(\psi)$
$[\ldots]$	$[\ldots]$
$\Box\psi_1$	$\Phi_i^0(\psi_1) \cap x_i^{\Box\psi_1}$
$\Box^-\psi_1$	$\Phi_i^0(\psi_1) \cap \Phi_{i-1}(\Box^-\psi_1)$
$\Diamond\psi_1$	$\Phi_i^0(\psi_1) \cup x_i^{\Diamond\psi_1}$
$\Diamond^-\psi_1$	$\Phi_i^0(\psi_1) \cup \Phi_{i-1}(\Diamond^-\psi_1)$

Table 3.4: $\Phi_i^0(\psi)$ with $\Box, \Box^-, \Diamond$ and $\Diamond^-$

$\mathsf{Sub}(\phi)$ denotes the set of all TQs occurring as temporal subqueries in ϕ (including ϕ itself).

Theorem 3.1.4. *The extension of the algorithm from [1] by $\Box, \Box^-, \Diamond$ and $\Diamond^-$ preserves correctness and boundedness.*

Proof. To prove that the correctness and boundedness of the algorithm is preserved, the necessary cases are added to the corresponding proofs from [1].

Lemma 3.1.5 (ct. Lemma 6.3 in [1])**.** *The function Φ_0 for $\Box, \Box^-, \Diamond$ and $\Diamond^-$ is correct for 0.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$ for all $n \geq 0$.
If $\psi = \Box^-\psi_1$ or $\psi = \Diamond^-\psi_1$, then

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{eval}^n(\Phi_0(\psi_1)).$$

This is by induction equal to $\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0)$ which then is, as shown in Proposition 3.1.2, equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
If $\psi = \Box\psi_1$, then

$$\begin{aligned} \mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_1)) \cap \mathsf{eval}^n(x_0^\psi) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \Delta^{\mathsf{N_V}} & \text{if } n = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0) \end{aligned}$$

If $\psi = \Diamond\psi_1$, then

$$\begin{aligned} \mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_1)) \cup \mathsf{eval}^n(x_0^\psi) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cup \left\{ \begin{array}{ll} \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \emptyset & \text{if } n = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0) \end{aligned}$$

□

Lemma 3.1.6 (ct. Lemma 6.4 in [1])**.** *If Φ_{i-1} for $\Box, \Box^-, \Diamond$ and $\Diamond^-$ is correct for i-1, then Φ_i^0 for $\Box, \Box^-, \Diamond$ and $\Diamond^-$ is correct for i.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_i^0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$ for all $n \geq i$. This is shown for two cases here. The other cases can be found in Appendix C.

If $\psi = \Box^- \psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(\Phi_{i-1}(\psi)) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i-1) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \Diamond \psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cup \mathsf{eval}^n(x_i^\psi) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cup \left\{ \begin{array}{ll} \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i \\ \emptyset & \text{if } n = i \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

□

Lemma 3.1.7 (ct. Lemma 6.5 in [1])**.** *If Φ_{i-1} for $\Box, \Box^-, \Diamond$ and $\Diamond^-$ is correct for i-1 and (i-1)-bounded, then we can construct a function $\Phi_i : \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ for $\Box, \Box^-, \Diamond$ and $\Diamond^-$ that is correct for i and i-bounded.*

Proof. The function $\mathsf{update}(x_{i-1}^{\psi^j})$, introduced in [1], needs to be extended, before it then can be shown for all $n \geq i$ that $\mathsf{eval}^n(x_{i-1}^{\psi^j})$ is still equal to $\mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j}))$. After considering the new operators, $\mathsf{update}(x_{i-1}^{\psi^j})$ looks like this:

$$\mathsf{update}(x_{i-1}^{\psi^j}) := \left\{ \begin{array}{ll} \Phi_i^{j-1}(\psi_1) & \text{if } \psi^j = \bigcirc\psi_1 \text{ or } \psi^j = \bullet\psi_1 \\ \Phi_i^{j-1}(\psi^j) & \text{if } \psi^j = \psi_1 \mathsf{U} \psi_2 \text{ or } \psi^j = \Box\psi_1 \text{ or } \psi^j = \Diamond\psi_1 \end{array} \right\}$$

For $\psi^j = \Box\psi_1$ and $\psi^j = \Diamond\psi_1$, by definition

$$\mathsf{eval}^n(x_{i-1}^{\psi^j}) = \mathsf{Ans}(\psi^j, \mathfrak{I}^{(n)}, i).$$

Since Φ_i^{j-1} is correct for i, this is the same set as

$$\mathsf{eval}^n(\Phi_i^{j-1}(\psi^j)) = \mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j})).$$

It remains to show i-boundedness of $\Phi_i = \Phi_i^k$. In [1] this is again proven by induction on j. It therefore suffices to add the missing cases. It is enough to show that $\mathsf{update}(x_{i-1}^{\psi^j})$ contains only variables from $\mathsf{Var}_i^{\psi^j}$.

If $\psi^j = \Box\psi_1$ or $\psi^j = \Diamond\psi_1$, then $\mathsf{update}(x_{i-1}^{\psi^j}) = \Phi_i^{j-1}(\psi^j)$. Since Φ_i^{j-1} differs from Φ_i^0 only in the replacement of some variables with index $i-1$,

$$\Phi_i^{j-1}(\psi^j) = \Phi_i^{j-1}(\psi_1) \cap x_i^{\psi^j}$$

or

$$\Phi_i^{j-1}(\psi^j) = \Phi_i^{j-1}(\psi_1) \cup x_i^{\psi^j}, \text{ respectively.}$$

By the induction hypothesis $\Phi_i^{j-1}(\psi_1)$ contains only variables from $\mathsf{Var}_i^{\psi_1} = \mathsf{Var}_i^{\psi^j} \setminus \{x_i^{\psi^j}\}$ and $\mathsf{Var}_{i-1}^{\psi_1} \cap \{x_{i-1}^{\psi^j}, \dots, x_{i-1}^{\psi^k}\}$. Since every variable $x_{i-1}^{\psi'} \in \mathsf{Var}_{i-1}^{\psi_1}$ must satisfy $\psi' \in \mathsf{FSub}(\psi_1)$ the second set $\mathsf{Var}_{i-1}^{\psi_1} \cap \{x_{i-1}^{\psi^j}, ..., x_{i-1}^{\psi^k}\}$ is empty. This follows from the total order $\psi^1 \prec \cdots \prec \psi^k$ on the set $\mathsf{FSub}(\phi) = \{\psi^1, \dots, \psi^k\}$ presented in [1], i.e. $\psi' \in \mathsf{FSub}(\psi^j) \setminus \{\psi^j\}$, and thus $\psi' \prec \psi^j$. $\Box$

This concludes the proof of Theorem 3.1.4. $\Box$

3.2 Filter Operator

Second, the algorithm presented in [1] is extended by a filter operator. This filter operator $f[\phi]$ denotes that the filter query f is applied to the result of ϕ. Since $\mathcal{Q}$-queries are not further specified, it is impossible to determine at which position in f the result of ϕ should be inserted. However, it is known that the implementation of the BHE [4] is based on SQL. Therefore it can be defined that f is an SQL statement and the result of ϕ is inserted at "SELECT * FROM ϕ".

The semantics of TQs from Definition 3.1.1 are extended as follows:

Definition 3.2.1 (semantics of temporal queries with a filter operator ct. Definition 3.3 in [1])**.** Let ϕ be a TQ, $\mathfrak{I} = (I_i)_{0 \le i \le n}$ a sequence of interpretations over a common domain, $\mathfrak{a} : \mathsf{FVar}(\phi) \to \mathsf{N_C}$ a variable assignment, f be an SQL statement, and i be an integer with $0 \le i \le n$. The *satisfaction relation* $\mathfrak{I}, i \models \mathfrak{a}(\phi)$ is defined by induction on the structure of ϕ as follows:

ϕ	$\mathfrak{I}, i \models \mathfrak{a}(\phi)$ iff
$[\dots]$	$[\dots]$
$f[\phi]$	$I_i \models \mathfrak{a}(f)$ and $\mathfrak{I}, i \models \mathfrak{a}(\phi)$

Table 3.5: Satisfaction relation of temporal queries with a filter operator

Now Query 2.2 can be rewritten as follows:

“SELECT * FROM ($\bigcirc^{-}$(SELECT *marke*, AVG(*price*) AS *price*1 FROM *autos*) $\wedge$ SELECT *marke*, AVG(*price*) AS *price*2 FROM *autos*) WHERE *price*1 < *price*2”.

The semantics of the filter operator can now be used to extend the algorithm specified in [1]. The functions $\mathsf{eval}^n : \mathsf{AT}^n_\phi \rightarrow 2^{\Delta^{\mathsf{N_V}}}, n \geq 0$ in [1] do not have to be extended.

The function $\Phi_0(\psi) : \mathsf{Sub}(\phi) \rightarrow \mathsf{AT}^0_\phi$ in [1] has to be extended as follows:

ψ	$\Phi_0(\psi)$
$[\ldots]$	$[\ldots]$
$f[\psi_1]$	$\mathsf{Ans}(f[\Phi_0(\psi_1)], I_0)$

Table 3.6: $\Phi_0(\psi)$ with a filter operator

The function $\Phi^0_i(\psi) : \mathsf{Sub}(\phi) \rightarrow \mathsf{AT}^i_\phi,\ i > 0$ in [1] has to be extended as follows:

ψ	$\Phi_0(\psi)$
$[\ldots]$	$[\ldots]$
$f[\psi_1]$	$\mathsf{Ans}(f[\Phi^0_i(\psi_1)], I_i)$

Table 3.7: $\Phi^0_i(\psi)$ with a filter operator

Theorem 3.2.2. *Extending the algorithm from [1] by a filter operator preserves correctness and boundedness.*

Proof. To prove that the correctness and boundedness of the algorithm is preserved, the necessary cases are added to the corresponding proofs from [1].

Lemma 3.2.3 (ct. Lemma 6.3 in [1])**.** *The function Φ_0 with a filter operator is correct for 0.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$ for all $n \geq 0$.
If $\psi = f[\psi_1]$, then

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{Ans}(f[\Phi_0(\psi_1)], I_0) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0).$$

□

Lemma 3.2.4 (ct. Lemma 6.4 in [1])**.** *If Φ_{i-1} with a filter operator is correct for i-1, then Φ_i^0 with a filter operator is correct for i.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_i^0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$ for all $n \geq i$.
If $\psi = f[\psi_1]$, then

$$\mathsf{eval}^n(\Phi_i^0(\psi)) = \mathsf{Ans}(f[\Phi_i^0(\psi_1)], I_i) = \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i).$$

□

Lemma 3.2.5 (ct. Lemma 6.5 in [1])**.** *If Φ_{i-1} with a filter operator is correct for i-1 and (i-1)-bounded, then we can construct a function $\Phi_i : \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ with a filter operator that is correct for i and i-bounded.*

Proof. Since $f[\psi_1]$ introduces no new variables, this follows directly from Lemma 6.5 in [1]. □

This concludes the proof of Theorem 3.2.2. □

3.3 Metric Temporal Operators

Third, the algorithm presented in [1] is extended by MTOs. For $\bigcirc\phi$, $\bullet\phi$, $\bigcirc^-\phi$ and $\bullet^-\phi$ the MTOs simplify composition, e.g. $\bigcirc_3\phi = \bigcirc(\bigcirc(\bigcirc\phi))$. For $\Box\phi$, $\Box^-\phi$, $\Diamond\phi$, $\Diamond^-\phi$, $\phi_1\mathsf{U}\phi_2$, and $\phi_1\mathsf{S}\phi_2$ MTOs restrict the number of time points, e.g. $\Box_5\phi$ (for 5 time points) or $\Diamond_5\phi$ (some time in 5 time points).

The semantics of TQs from Definition 3.1.1 are extended as follows:

Definition 3.3.1 (semantics of temporal queries with metric temporal operators ct. Definition 3.3 in [1])**.** Let ϕ be a TQ, $\mathfrak{I} = (I_i)_{0\leq i\leq n}$ a sequence of interpretations over a common domain, $\mathfrak{a} : \mathsf{FVar}(\phi) \to \mathsf{N_C}$ a variable assignment, i be an integer with $0 \leq i \leq n$, and p be an integer with $p \geq 0$.

The *satisfaction relation* $\mathfrak{I}, i \models \mathfrak{a}(\phi)$ is defined by induction on the structure of ϕ as follows:

ϕ	$\mathfrak{I}, i \models \mathfrak{a}(\phi)$ iff
$[\ldots]$	$[\ldots]$
$\bigcirc_p \phi_1$	$i + p \leq n$ and $\mathfrak{I}, i + p \models \mathfrak{a}(\phi_1)$
$\bullet_p \phi_1$	$i + p \leq n$ implies $\mathfrak{I}, i + p \models \mathfrak{a}(\phi_1)$
$\bigcirc_p^- \phi_1$	$i - p \geq 0$ and $\mathfrak{I}, i - p \models \mathfrak{a}(\phi_1)$
$\bullet_p^- \phi_1$	$i - p \geq 0$ implies $\mathfrak{I}, i - p \models \mathfrak{a}(\phi_1)$
$\Box_p \phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $i \leq k \leq \mathsf{min}(i + p, n)$
$\Box_p^- \phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $\mathsf{max}(i - p, 0) \leq k \leq i$
$\Diamond_p \phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $i \leq k \leq \mathsf{min}(i + p, n)$
$\Diamond_p^- \phi_1$	$\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $\mathsf{max}(i - p, 0) \leq k \leq i$
$\phi_1 \mathsf{U}_p \phi_2$	there is k, $i \leq k \leq \mathsf{min}(i + p, n)$, with $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all $j, i \leq j < k$
$\phi_1 \mathsf{S}_p \phi_2$	there is k, $\mathsf{max}(i - p, 0) \leq k \leq i$, with $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all $j, k < j \leq i$

Table 3.8: Satisfaction relation of temporal queries with metric temporal operators

$\mathfrak{a}_{\phi_1}$ denotes the restriction of a variable assignment $\mathfrak{a} : \mathsf{FVar}(\phi) \rightarrow \mathsf{N_C}$ to $\mathsf{FVar}(\phi_1)$ for a subquery ϕ_1 of ϕ.

Now Query 2.3 can be rewritten as follows:

"$\Diamond_6^-$(SELECT *url* FROM *autos* WHERE *price* > 1000000)"

Proposition 3.3.2 (ct. Proposition 3.4 in [1]). *For* $\mathfrak{a} : \mathsf{FVar}(\phi) \rightarrow \mathsf{N_C}$, $0 \leq i \leq n$ *and* $p = 0$, $\mathfrak{I}, i \models \mathfrak{a}(\phi)$ *iff* $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or* $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$, *respectively.*

Proof. To prove the above proposition, three equivalences are demonstrated here. The missing cases work similarly and can be found in Appendix C. The proof works mainly based on semantics.

- $\bigcirc_0 \phi_1 \equiv \phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\bigcirc_0 \phi_1) \tag{3.11}$$
$$\Leftrightarrow i + 0 \leq n \text{ and } \mathfrak{I}, i + 0 \models \mathfrak{a}(\phi_1) \tag{3.12}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \tag{3.13}$$

- $\Box_0^- \phi_1 \equiv \phi_1$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\Box_0^- \phi_1) && (3.14)\\
\Leftrightarrow &\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, \mathsf{max}(i-0,0) \leq k \leq i && (3.15)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) && (3.16)
\end{aligned}$$

- $\phi_1 \mathsf{U}_0 \phi_2 \equiv \phi_2$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{U}_0 \phi_2) && (3.17)\\
\Leftrightarrow &\text{there is } k, i \leq k \leq \mathsf{min}(i+0,n), \text{ with } \mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and} && (3.18)\\
&\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, i \leq j < k &&\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) && (3.19)
\end{aligned}$$

□

There are again equivalences similar to Proposition 3.1.2.

- $\bigcirc_p \phi_1$ is equivalent to $\bigcirc\bigcirc_{p-1}\phi_1$,
- $\bullet_p \phi_1$ is equivalent to $\bullet\bullet_{p-1}\phi_1$,
- $\bigcirc_p^- \phi_1$ is equivalent to $\bigcirc^- \bigcirc_{p-1}^- \phi_1$,
- $\bullet_p^- \phi_1$ is equivalent to $\bullet^- \bullet_{p-1}^- \phi_1$,
- $\Box_p \phi_1$ is equivalent to $\phi_1 \wedge \bullet \Box_{p-1} \phi_1$,
- $\Box_p^- \phi_1$ is equivalent to $\phi_1 \wedge \bullet^- \Box_p^- \phi_1$,
- $\Diamond_p \phi_1$ is equivalent to $\phi_1 \vee \bigcirc \Diamond_p \phi_1$,
- $\Diamond_p^- \phi_1$ is equivalent to $\phi_1 \vee \bigcirc^- \Diamond_p^- \phi_1$,
- $\phi_1 \mathsf{U}_p \phi_2$ is equivalent to $\phi_2 \vee (\phi_1 \wedge \bigcirc(\phi_1 \mathsf{U}_{p-1} \phi_2))$, and
- $\phi_1 \mathsf{S}_p \phi_2$ is equivalent to $\phi_2 \vee (\phi_1 \wedge \bigcirc^-(\phi_1 \mathsf{S}_{p-1} \phi_2))$.

Thus, at the last time point

- $\bigcirc_p \phi_1$ does not have any answers because $\bigcirc\bigcirc_{p-1}\phi_1$ does not have any answers,
- $\bullet_p \phi_1$ is tautological because $\bullet\bullet_{p-1}\phi_1$ is tautological,

- $\Box_p\phi_1$ is equivalent to ϕ_1 because $\bullet\Box_{p-1}\phi_1$ is tautological,
- $\Diamond_p\phi_1$ is equivalent to ϕ_1 because $\bigcirc\Diamond_p\phi_1$ does not have any answers, and
- $\phi_1\mathsf{U}_p\phi_2$ is equivalent to ϕ_2 because $\bigcirc(\phi_1\mathsf{U}_{p-1}\phi_2)$ does not have any answers,

and at the first time point

- $\bigcirc_p^-\phi_1$ does not have any answers because $\bigcirc^-\bigcirc_{p-1}^-\phi_1$ does not have any answers,
- $\bullet_p^-\phi_1$ is tautological because $\bullet^-\bullet_{p-1}^-\phi_1$ is tautological,
- $\Box_p^-\phi_1$ is equivalent to ϕ_1 because $\bullet^-\Box_p^-\phi_1$ is tautological,
- $\Diamond_p^-\phi_1$ is equivalent to ϕ_1 because $\bigcirc^-\Diamond_p^-\phi_1$ does not have any answers, and
- $\phi_1\mathsf{S}_p\phi_2$ is equivalent to ϕ_2 because $\bigcirc^-(\phi_1\mathsf{S}_{p-1}\phi_2)$ does not have any answers.

Proposition 3.3.3 (ct. Proposition 3.4 in [1]). *For* $\mathfrak{a} : \mathsf{FVar}(\phi) \to \mathsf{N_C}$, $0 \le i \le n$ *and* $p > 0$,

1. $\mathfrak{I}, i \models \mathfrak{a}(\bigcirc_p\phi_1)$ *iff*
   - $i < n$ *and* $\mathfrak{I}, i+1 \models \mathfrak{a}(\bigcirc_{p-1}\phi_1)$
2. $\mathfrak{I}, i \models \mathfrak{a}(\bullet_p\phi_1)$ *iff*
   - $i < n$ *implies* $\mathfrak{I}, i+1 \models \mathfrak{a}(\bullet_{p-1}\phi_1)$
3. $\mathfrak{I}, i \models \mathfrak{a}(\bigcirc_p^-\phi_1)$ *iff*
   - $i > 0$ *and* $\mathfrak{I}, i-1 \models \mathfrak{a}(\bigcirc_{p-1}^-\phi_1)$
4. $\mathfrak{I}, i \models \mathfrak{a}(\bullet_p^-\phi_1)$ *iff*
   - $i > 0$ *implies* $\mathfrak{I}, i-1 \models \mathfrak{a}(\bullet_{p-1}^-\phi_1)$
5. $\mathfrak{I}, i \models \mathfrak{a}(\Box_p\phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *and*
   - $i < n$ *implies* $\mathfrak{I}, i+1 \models \mathfrak{a}(\Box_{p-1}\phi_1)$

6. $\mathfrak{I}, i \models \mathfrak{a}(\Box^-_p \phi_1)$ *iff*

- $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *and*
- $i > 0$ *implies* $\mathfrak{I}, i-1 \models \mathfrak{a}(\Box^-_{p-1}\phi_1)$

7. $\mathfrak{I}, i \models \mathfrak{a}(\Diamond_p \phi_1)$ *iff*

- $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or*
- $i < n$ *and* $\mathfrak{I}, i+1 \models \mathfrak{a}(\Diamond_{p-1}\phi_1)$

8. $\mathfrak{I}, i \models \mathfrak{a}(\Diamond^-_p \phi_1)$ *iff*

- $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or*
- $i > 0$ *and* $\mathfrak{I}, i-1 \models \mathfrak{a}(\Diamond^-_{p-1}\phi_1)$

9. $\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{U}_p \phi_2)$ *iff*

- $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$ *or*
- $\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1)$ *and* $i < n$ *and* $\mathfrak{I}, i+1 \models \mathfrak{a}(\phi_1 \mathsf{U}_{p-1} \phi_2)$

10. $\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{S}_p \phi_2)$ *iff*

- $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ *or*
- $\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1)$ *and* $i > 0$ *and* $\mathfrak{I}, i-1 \models \mathfrak{a}(\phi_1 \mathsf{S}_{p-1} \phi_2)$

Proof. To prove the above proposition, three equivalences are demonstrated here. The missing cases work similarly and can be found in Appendix C. The proof works mainly based on semantics.

1. $\bigcirc_p \phi_1 \equiv \bigcirc \bigcirc_{p-1} \phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\bigcirc_p \phi_1) \tag{3.20}$$
$$\Leftrightarrow i + p \leq n \text{ and } \mathfrak{I}, i + p \models \mathfrak{a}(\phi_1) \tag{3.21}$$
$$\Leftrightarrow i < n \text{ and } (i+1) + (p-1) \leq n \text{ and } \mathfrak{I}, (i+1) + (p-1) \models \mathfrak{a}(\phi_1) \tag{3.22}$$
$$\Leftrightarrow i < n \text{ and } \mathfrak{I}, i+1 \models \mathfrak{a}(\bigcirc_{p-1} \phi_1) \tag{3.23}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\bigcirc \bigcirc_{p-1} \phi_1) \tag{3.24}$$

6. $\Box_p^- \phi_1 \equiv \phi_1 \wedge \bullet^- \Box_p^- \phi_1$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\Box_p^- \phi_1) && (3.25)\\
\Leftrightarrow &\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, \mathsf{max}(i-p, 0) \le k \le i && (3.26)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i > 0 \text{ implies} && (3.27)\\
&\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, \mathsf{max}((i-1)-(p-1), 0) \le k \le i-1)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i > 0 \text{ implies } \mathfrak{I}, i-1 \models \mathfrak{a}(\Box_{p-1}^- \phi_1)) && (3.28)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \wedge \bullet^- \Box_{p-1}^- \phi_1) && (3.29)
\end{aligned}$$

(3.27) is equivalent to (3.26) because

- in case $i > 0$, the query needs to be satisfied now, at time point i, and at past time points k, $\mathsf{max}((i-1)-(p-1), 0) \le k \le i-1$, to be satisfied. Since $i > 0$ is true, the satisfaction of past time points depends solely on the second part of the "implies"-statement; and
- in case $i = 0$, the query needs to be satisfied now, at time point i, to be satisfied. There are no past time points k, $\mathsf{max}((i-1)-(p-1), 0) \le k \le 0-1$. Since $i = 0$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $\mathsf{max}(i-p, 0) \le k \le 0$, and $i > 0$ is not true, thus the "implies"-statement does not affect satisfaction.

9. $\phi_1 \mathsf{U}_p \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge \bigcirc(\phi_1 \mathsf{U}_{p-1} \phi_2))$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{U}_p \phi_2) && (3.30)\\
\Leftrightarrow &\text{there is } k, i \le k \le \mathsf{min}(i+p, n), \text{ with } \mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and} && (3.31)\\
&\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, i \le j < k\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ or } (\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ and } (i < n \text{ and} && (3.32)\\
&\text{there is } k, i+1 \le k \le \mathsf{min}((i+1)+(p-1), n), \text{ with}\\
&\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and } \mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, i+1 \le j < k))\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ or } (\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ and } (i < n \text{ and} && (3.33)\\
&\mathfrak{I}, i+1 \models \mathfrak{a}(\phi_1 \mathsf{U}_{p-1} \phi_2)))\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_2 \vee (\phi_1 \wedge \bigcirc(\phi_1 \mathsf{U}_{p-1} \phi_2))) && (3.34)
\end{aligned}$$

(3.32) is equivalent to (3.31) because

- in case $i < n$, either ϕ_2 needs to be satisfied now, at time point i, or ϕ_1 needs to be satisfied now, at time point i, and there needs to be a future time point k, $i+1 \le k \le \mathsf{min}((i+1)+(p-1), n)$, where ϕ_2 is satisfied and ϕ_1 is satisfied for all time points j, $i+1 \le j < k$, for the query to be satisfied.

- in case $i = n$, ϕ_2 needs to be satisfied now, at time point i, for the query to be satisfied. There are no future time points k, $n+1 \leq k \leq \mathsf{min}((i+1)+(p-1), n)$. Since $i = n$ is true, $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$ is equivalent to there is k, $n \leq k \leq \mathsf{min}(i+p, n)$, with $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all $j, n \leq j < k$, and $i < n$ is not true, thus the "or"-statement does not affect satisfaction.

$\square$

The semantics of the MTOs can now be used to extend the algorithm specified in [1].

The functions $\mathsf{eval}^n : \mathsf{AT}^n_\phi \to 2^{\Delta^{\mathsf{N_V}}}, n \geq 0$ in [1] have then to be extended as follows:

α	$\mathsf{eval}^n(\alpha)$
$[\ldots]$	$[\ldots]$
$x_j^{\bigcirc_p \psi_1}$ with $j < n$	$\mathsf{Ans}(\bigcirc_{p-1}\psi_1, \mathfrak{I}^{(n)}, j+1)$
$x_j^{\bullet_p \psi_1}$ with $j < n$	$\mathsf{Ans}(\bullet_{p-1}\psi_1, \mathfrak{I}^{(n)}, j+1)$
$x_j^{\Box_p \psi_1}$ with $j < n$	$\mathsf{Ans}(\Box_{p-1}\psi_1, \mathfrak{I}^{(n)}, j+1)$
$x_j^{\Diamond_p \psi_1}$ with $j < n$	$\mathsf{Ans}(\Diamond_{p-1}\psi_1, \mathfrak{I}^{(n)}, j+1)$
$x_j^{\psi_1 \mathsf{U}_p \psi_2}$ with $j < n$	$\mathsf{Ans}(\psi_1 \mathsf{U}_{p-1} \psi_2, \mathfrak{I}^{(n)}, j+1)$
$x_n^{\bigcirc_p \psi_1}$	$\emptyset$
$x_n^{\bullet_p \psi_1}$	$\Delta^{\mathsf{N_V}}$
$x_n^{\Box_p \psi_1}$	$\Delta^{\mathsf{N_V}}$
$x_n^{\Diamond_p \psi_1}$	$\emptyset$
$x_n^{\psi_1 \mathsf{U}_p \psi_2}$	$\emptyset$

Table 3.9: $\mathsf{eval}^n(\alpha)$ with metric temporal operators

The function $\Phi_0(\psi) : \mathsf{Sub}(\phi) \rightarrow \mathsf{AT}^0_\phi$ in [1] has to be extended as follows:

ψ	$\Phi_0(\psi)$
$[\ldots]$	$[\ldots]$
$\bigcirc_p \psi_1$ with $p > 0$	$x_0^{\bigcirc_p \psi_1}$
$\bullet_p \psi_1$ with $p > 0$	$x_0^{\bullet_p \psi_1}$
$\bigcirc_p^- \psi_1$ with $p > 0$	$\emptyset$
$\bullet_p^- \psi_1$ with $p > 0$	$\Delta^{\mathsf{N_V}}$
$\Box_p \psi_1$ with $p > 0$	$\Phi_0(\psi_1) \cap x_0^{\Box_p \psi_1}$
$\Box_p^- \psi_1$	$\Phi_0(\psi_1)$
$\Diamond_p \psi_1$ with $p > 0$	$\Phi_0(\psi_1) \cup x_0^{\Diamond_p \psi_1}$
$\Diamond_p^- \psi_1$	$\Phi_0(\psi_1)$
$\psi_1 \mathsf{U}_p \psi_2$ with $p > 0$	$\Phi_0(\psi_2) \cup (\Phi_0(\psi_1) \cap x_0^{\psi_1 \mathsf{U}_p \psi_2})$
$\psi_1 \mathsf{S}_p \psi_2$	$\Phi_0(\psi_2)$
$\bigcirc_0 \psi_1$	$\Phi_0(\psi_1)$
$\bullet_0 \psi_1$	$\Phi_0(\psi_1)$
$\bigcirc_0^- \psi_1$	$\Phi_0(\psi_1)$
$\bullet_0^- \psi_1$	$\Phi_0(\psi_1)$
$\Box_0 \psi_1$	$\Phi_0(\psi_1)$
$\Diamond_0 \psi_1$	$\Phi_0(\psi_1)$
$\psi_1 \mathsf{U}_0 \psi_2$	$\Phi_0(\psi_2)$

Table 3.10: $\Phi_0(\psi)$ with metric temporal operators

The function $\Phi_i^0(\psi) : \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i,\ i > 0$ in [1] has to be extended as follows:

ψ	$\Phi_i^0(\psi)$
$[\ldots]$	$[\ldots]$
$\bigcirc_p\psi_1$ with $p > 0$	$x_i^{\bigcirc_p\psi_1}$
$\bullet_p\psi_1$ with $p > 0$	$x_i^{\bullet_p\psi_1}$
$\bigcirc_p^-\psi_1$ with $p > 0$	$\Phi_{i-1}(\bigcirc_{p-1}^-\psi_1)$
$\bullet_p^-\psi_1$ with $p > 0$	$\Phi_{i-1}(\bullet_{p-1}^-\psi_1)$
$\Box_p\psi_1$ with $p > 0$	$\Phi_i^0(\psi_1) \cap x_i^{\Box_p\psi_1}$
$\Box_p^-\psi_1$ with $p > 0$	$\Phi_i^0(\psi_1) \cap \Phi_{i-1}(\Box_{p-1}^-\psi_1)$
$\Diamond_p\psi_1$ with $p > 0$	$\Phi_i^0(\psi_1) \cup x_i^{\Diamond_p\psi_1}$
$\Diamond_p^-\psi_1$ with $p > 0$	$\Phi_i^0(\psi_1) \cup \Phi_{i-1}(\Diamond_{p-1}^-\psi_1)$
$\psi_1\mathsf{U}_p\psi_2$ with $p > 0$	$\Phi_i^0(\psi_2) \cup (\Phi_i^0(\psi_1) \cap x_i^{\psi_1\mathsf{U}_p\psi_2})$
$\psi_1\mathsf{S}_p\psi_2$ with $p > 0$	$\Phi_i^0(\psi_2) \cup (\Phi_i^0(\psi_1) \cap \Phi_{i-1}(\psi_1\mathsf{S}_{p-1}\psi_2))$
$\bigcirc_0\psi_1$	$\Phi_i^0(\psi_1)$
$\bullet_0\psi_1$	$\Phi_i^0(\psi_1)$
$\bigcirc_0^-\psi_1$	$\Phi_i^0(\psi_1)$
$\bullet_0^-\psi_1$	$\Phi_i^0(\psi_1)$
$\Box_0\psi_1$	$\Phi_i^0(\psi_1)$
$\Box_0^-\psi_1$	$\Phi_i^0(\psi_1)$
$\Diamond_0\psi_1$	$\Phi_i^0(\psi_1)$
$\Diamond_0^-\psi_1$	$\Phi_i^0(\psi_1)$
$\psi_1\mathsf{U}_0\psi_2$	$\Phi_i^0(\psi_2)$
$\psi_1\mathsf{S}_0\psi_2$	$\Phi_i^0(\psi_2)$

Table 3.11: $\Phi_i^0(\psi)$ with metric temporal operators

$\mathsf{Sub}(\phi)$ now includes all queries $\bigcirc_m\psi_1$, $\bullet_m\psi_1$, $\bigcirc_m^-\psi_1$, $\bullet_m^-\psi_1$, $\Box_m\psi_1$, $\Box_m^-\psi_1$, $\Diamond_m\psi_1$, $\Diamond_m^-\psi_1$, $\psi_1\mathsf{U}_m\psi_2$ and $\psi_1\mathsf{S}_m\psi_2$ with m, $0 \le m \le p$. $\mathsf{FSub}(\phi)$, the subset of queries from $\mathsf{Sub}(\phi)$ that start with a future operator, is extended accordingly. It is noteworthy, that even though the sets denoted by $\mathsf{Sub}(\phi)$ and $\mathsf{FSub}(\phi)$ for ϕ with MTOs are different from the sets for ϕ without MTOs, they are the same size, e.g. for Query 2.3 without MTOs $\mathsf{Sub}(\phi) = \{\phi, \bigcirc^-\bigcirc^-\bigcirc^-\bigcirc^-\bigcirc^-\psi, \bigcirc^-\bigcirc^-\bigcirc^-\bigcirc^-\psi, \bigcirc^-\bigcirc^-\bigcirc^-\psi, \bigcirc^-\bigcirc^-\psi, \bigcirc^-\psi, \psi\}$ and with MTOs $\mathsf{Sub}(\phi) = \{\phi, \Diamond_5^-\psi, \Diamond_4^-\psi, \Diamond_3^-\psi, \Diamond_2^-\psi, \Diamond_1^-\psi, \Diamond_0^-\psi\}$.

Theorem 3.3.4. *Extending the algorithm from [1] by metric temporal operators preserves correctness and boundedness.*

Proof. To prove that the correctness and boundedness of the algorithm is preserved, the necessary cases are added to the corresponding proofs from [1].

Lemma 3.3.5 (ct. Lemma 6.3 in [1]). *The function Φ_0 with metric temporal operators is correct for 0.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$ for all $n \geq 0$. The missing cases can be found in Appendix C.
If $\psi = \bigcirc_0 \psi_1$, $\psi = \newmoon_0 \psi_1$, $\psi = \bigcirc_0^- \psi_1$, $\psi = \newmoon_0^- \psi_1$, $\psi = \Box_0 \psi_1$, $\psi = \Box_0^- \psi_1$, $\psi = \Diamond_0 \psi_1$ or $\psi = \Diamond_0^- \psi_1$, then

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{eval}^n(\Phi_0(\psi_1)).$$

This is by induction equal to $Ans(\psi_1, \mathfrak{I}^{(n)}, 0)$ which then is, as shown in Proposition 3.3.2, equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
If $\psi = \psi_1 \mathsf{U}_0 \psi_2$ or $\psi = \psi_1 \mathsf{S}_0 \psi_2$, then

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{eval}^n(\Phi_0(\psi_2)).$$

This is by induction equal to $Ans(\psi_2, \mathfrak{I}^{(n)}, 0)$ which then is, as shown in Proposition 3.3.2, equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
If $\psi = \bigcirc_p^- \psi_1$, $\psi = \newmoon_p^- \psi_1$, $\psi = \Box_p^- \psi_1$ or $\psi = \Diamond_p^- \psi_1$ and $p > 0$, then

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{eval}^n(\Phi_0(\psi_1)).$$

This is by induction equal to $\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0)$ which then is, as shown in Proposition 3.3.3, equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
If $\psi = \bigcirc_p \psi_1$ and $p > 0$, then

$$\begin{aligned} \mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(x_0^{\bigcirc_p \psi_1}) \\ &= \left\{ \begin{array}{ll} \mathsf{Ans}(\bigcirc_{p-1}\psi_1, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \emptyset & \text{if } n = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0) \end{aligned}$$

If $\psi = \Box_p \psi_1$ and $p > 0$, then

$$\begin{aligned} \mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_1)) \cap \mathsf{eval}^n(x_0^{\Box_p \psi_1}) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\Box_{p-1}\psi_1, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \Delta^{\mathsf{N_V}} & \text{if } n = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0) \end{aligned}$$

If $\psi = \psi_1 \mathsf{U}_p \psi_2$ and $p > 0$, then

$$\begin{aligned}
\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_2)) \cup (\mathsf{eval}^n(\Phi_0(\psi_1)) \cap \mathsf{eval}^n(x_0^{\psi_1 \mathsf{U}_p \psi_2})) \\
&= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, 0) \cup (\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\psi_1 \mathsf{U}_{p-1} \psi_2, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \emptyset & \text{if } n = 0 \end{array} \right\}) \\
&= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)
\end{aligned}$$

□

Lemma 3.3.6 (ct. Lemma 6.4 in [1]). *If Φ_{i-1} with metric temporal operators is correct for i-1, then Φ_i^0 with metric temporal operators is correct for i.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_i^0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$ for all $n \geq i$. The missing cases can be found in Appendix C.

If $\psi = \bigcirc_0 \psi_1$, $\psi = \bullet_0 \psi_1$, $\psi = \bigcirc_0^- \psi_1$, $\psi = \bullet_0^- \psi_1$, $\psi = \Box_0 \psi_1$, $\psi = \Box_0^- \psi_1$, $\psi = \Diamond_0 \psi_1$ or $\psi = \Diamond_0^- \psi_1$, then

$$\begin{aligned}
\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \\
&= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \\
&= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)
\end{aligned}$$

If $\psi = \psi_1 \mathsf{U}_0 \psi_2$ or $\psi = \psi_1 \mathsf{S}_0 \psi_2$, then

$$\begin{aligned}
\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_2)) \\
&= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, i) \\
&= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)
\end{aligned}$$

If $\psi = \bigcirc_p \psi_1$ and $p > 0$, then

$$\begin{aligned}
\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(x_i^{\bigcirc_p \psi_1}) \\
&= \left\{ \begin{array}{ll} \mathsf{Ans}(\bigcirc_{p-1} \psi_1, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i \\ \emptyset & \text{if } n = i \end{array} \right\} \\
&= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)
\end{aligned}$$

If $\psi = \Box_p^- \psi_1$ and $p > 0$, then

$$\begin{aligned}
\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(\Phi_{i-1}(\Box_p^- \psi_1)) \\
&= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \mathsf{Ans}(\Box_{p-1}^- \psi_1, \mathfrak{I}^{(n)}, i-1) \\
&= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)
\end{aligned}$$

If $\psi = \psi_1 \mathsf{U}_p \psi_2$ and $p > 0$, then

$$\begin{aligned}
\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_2)) \cup (\mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(x_i^{\psi_1 \mathsf{U}_p \psi_2})) \\
&= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, i) \cup (\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\psi_1 \mathsf{U}_{p-1} \psi_2, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i \\ \emptyset & \text{if } n = i \end{array} \right\}) \\
&= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)
\end{aligned}$$

□

Lemma 3.3.7 (ct. Lemma 6.5 in [1])**.** *If Φ_{i-1} with metric temporal operators is correct for i-1 and (i-1)-bounded, then we can construct a function $\Phi_i : \mathsf{Sub}(\phi) \to \mathsf{AT}_\phi^i$ with metric temporal operators that is correct for i and i-bounded.*

Proof. The function $\mathsf{update}(x_{i-1}^{\psi^j})$, introduced in [1], needs to be extended, before it then can be shown for all $n \geq i$ that $\mathsf{eval}^n(x_{i-1}^{\psi^j})$ is still equal to $\mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j}))$. For operators with metric temporal operators, $\mathsf{update}(x_{i-1}^{\psi^j})$ looks like this:

$$\mathsf{update}(x_{i-1}^{\psi^j}) := \left\{ \begin{array}{ll} \Phi_i^{j-1}(\bigcirc_{p-1}\psi_1) & \text{if } \psi^j = \bigcirc_p \psi_1 \\ \Phi_i^{j-1}(\bullet_{p-1}\psi_1) & \text{if } \psi^j = \bullet_p \psi_1 \\ \Phi_i^{j-1}(\Box_{p-1}\psi_1) & \text{if } \psi^j = \Box_p \psi_1 \\ \Phi_i^{j-1}(\Diamond_{p-1}\psi_1) & \text{if } \psi^j = \Diamond_p \psi_1 \\ \Phi_i^{j-1}(\psi_1 \mathsf{U}_{p-1} \psi_2) & \text{if } \psi^j = \psi_1 \mathsf{U}_p \psi_2 \end{array} \right\}$$

By definition

$$\begin{gathered}
\mathsf{eval}^n(x_{i-1}^{\bigcirc_p \psi_1}) = \mathsf{Ans}(\bigcirc_{p-1}\psi_1, \mathfrak{I}^{(n)}, i), \\
\mathsf{eval}^n(x_{i-1}^{\bullet_p \psi_1}) = \mathsf{Ans}(\bullet_{p-1}\psi_1, \mathfrak{I}^{(n)}, i), \\
\mathsf{eval}^n(x_{i-1}^{\Box_p \psi_1}) = \mathsf{Ans}(\Box_{p-1}\psi_1, \mathfrak{I}^{(n)}, i), \\
\mathsf{eval}^n(x_{i-1}^{\Diamond_p \psi_1}) = \mathsf{Ans}(\Diamond_{p-1}\psi_1, \mathfrak{I}^{(n)}, i) \text{ and} \\
\mathsf{eval}^n(x_{i-1}^{\psi_1 \mathsf{U}_p \psi_2}) = \mathsf{Ans}(\psi_1 \mathsf{U}_{p-1} \psi_2, \mathfrak{I}^{(n)}, i).
\end{gathered}$$

Since Φ_i^{j-1} is correct for i, these are the same sets as

$$\mathsf{eval}^n(\Phi_i^{j-1}(\psi^j)) = \mathsf{eval}^n(\mathsf{update}(x_{i-1}^{\psi^j})).$$

It remains to show i-boundedness of $\Phi_i = \Phi_i^k$. In [1] this is again proven by induction on j. It therefore suffices to add the missing cases. It is enough to show that $\mathsf{update}(x_{i-1}^{\psi^j})$ contains only variables from $\mathsf{Var}_i^{\psi^j}$. Since Φ_i^{j-1} differs from Φ_i^0 only in the replacement of some variables with index $i-1$,

$$\begin{aligned}
&\Phi_i^{j-1}(\bigcirc_{p-1}\psi_1) = x_i^{\bigcirc_p\psi_1} \\
&\Phi_i^{j-1}(\bullet_{p-1}\psi_1) = x_i^{\bullet_p\psi_1} \\
&\Phi_i^{j-1}(\Box_{p-1}\psi_1) = \Phi_i^{j-1}(\psi_1) \cap x_i^{\Box_p\psi_1} \\
&\Phi_i^{j-1}(\Diamond_{p-1}\psi_1) = \Phi_i^{j-1}(\psi_1) \cup x_i^{\Diamond_p\psi_1} \\
&\Phi_i^{j-1}(\psi_1 \mathsf{U}_{p-1}\psi_2) = \Phi_i^{j-1}(\psi_2) \cup (\Phi_i^{j-1}(\psi_1) \cap x_i^{\psi_1\mathsf{U}_p\psi_2})
\end{aligned}$$

By the induction hypothesis each $\Phi_i^{j-1}(\psi_m), m = 1, 2$, contains only variables from $\mathsf{Var}_i^{\psi_m} = \mathsf{Var}_i^{\psi^j} \setminus \{x_i^{\psi^j}\}$ and $\mathsf{Var}_{i-1}^{\psi_m} \cap \{x_{i-1}^{\psi^j}, \ldots, x_{i-1}^{\psi^k}\}$. Since every variable $x_{i-1}^{\psi'} \in \mathsf{Var}_{i-1}^{\psi_1}$ must satisfy $\psi' \in \mathsf{FSub}(\psi_1)$ the second set $\mathsf{Var}_{i-1}^{\psi_1} \cap \{x_{i-1}^{\psi^j}, ..., x_{i-1}^{\psi^k}\}$ is empty. This follows from the total order $\psi^1 \prec \ldots \prec \psi^k$ on the set $\mathsf{FSub}(\phi) = \{\psi^1, \ldots, \psi^k\}$ presented in [1], i.e. $\psi' \in \mathsf{FSub}(\psi^j) \setminus \{\psi^j\}$, and thus $\psi' \prec \psi^j$. $\Box$

This concludes the proof of Theorem 3.3.4. $\Box$

3.4 Combined Extensions

In the following, only the extensions of the algorithm from [1] by a filter operator and MTOs will be discussed, as $\Box, \Box^-, \Diamond$ and $\Diamond^-$ can be seen as special cases of MTOs, where $p = n - i$ or $p = i$, respectively.

From the proofs of Theorem 3.2.2 and Theorem 3.3.4 it follows, that extending the algorithm from [1] by a filter operator and MTOs preserves correctness and boundedness.

In terms of expressiveness, only the filter operator adds new expressiveness. This has already been hinted at in Chapter 2, when introducing the MTOs and follows from Proposition 3.3.2 and Proposition 3.3.3. Since the filter operator is defined to be an SQL statement, the expressiveness added by introducing the filter operator is the same as an SQL statement on a single table.

However, the MTOs are still relevant, as they can reduce the size of queries significantly. A query without MTOs that is equivalent to, e.g. $\Box$ with a time limit t, consists of at least $2 * t$ operators, a $\wedge$ and a $\bullet$ for each time point up to the limit. Whereas, the corresponding query with MTOs consists of only one operator $\Box_t$ regardless of the size of t.

Lastly, it is feasible to implement the extensions as the main modification needed are case-distinctions in the implementations of $\Phi_0(\psi)$ and $\Phi_i^0(\psi)$ [4].

Chapter 4

Evaluation

After extending the implementation of the BHE [4], the queries introduced in Chapter 2 were evaluated at all 10 time points using this extended implementation [4]. This chapter summarizes how well the queries can be answered with the help of a BHE, regarding the size of the encoding, the time it takes to answer one query per time point, and the usefulness of the answers. The size of the encoding is compared to the size of the established approach of *system-versioned temporal tables* (SVTT) in SQL:2011 [11], which saves the complete history.

4.1 Evaluation of Practical Temporal Queries

The BHE [1] provides meaningful answers for all specified PTQs. It answers the query and returns what was asked for. The average time to answer one PTQ per time point is just under 233 milliseconds. For all PTQs, the BHE [1] stored fewer entries by the third time point at the latest. At the 10th time point on average only 16% of the entries were stored, compared to the SVTT [11] approach. For the examples from Chapter 2 the size of the encodings compared to the complete history can be seen in Figure 4.1.

Since it was possible to express all PTQs by past operators only, the PTQs show that the BHE [1] does provide meaningful answers to temporal queries, but they do not allow reliable conclusions to be drawn for the entire language of [1]. This is what the RTQs are intended for.

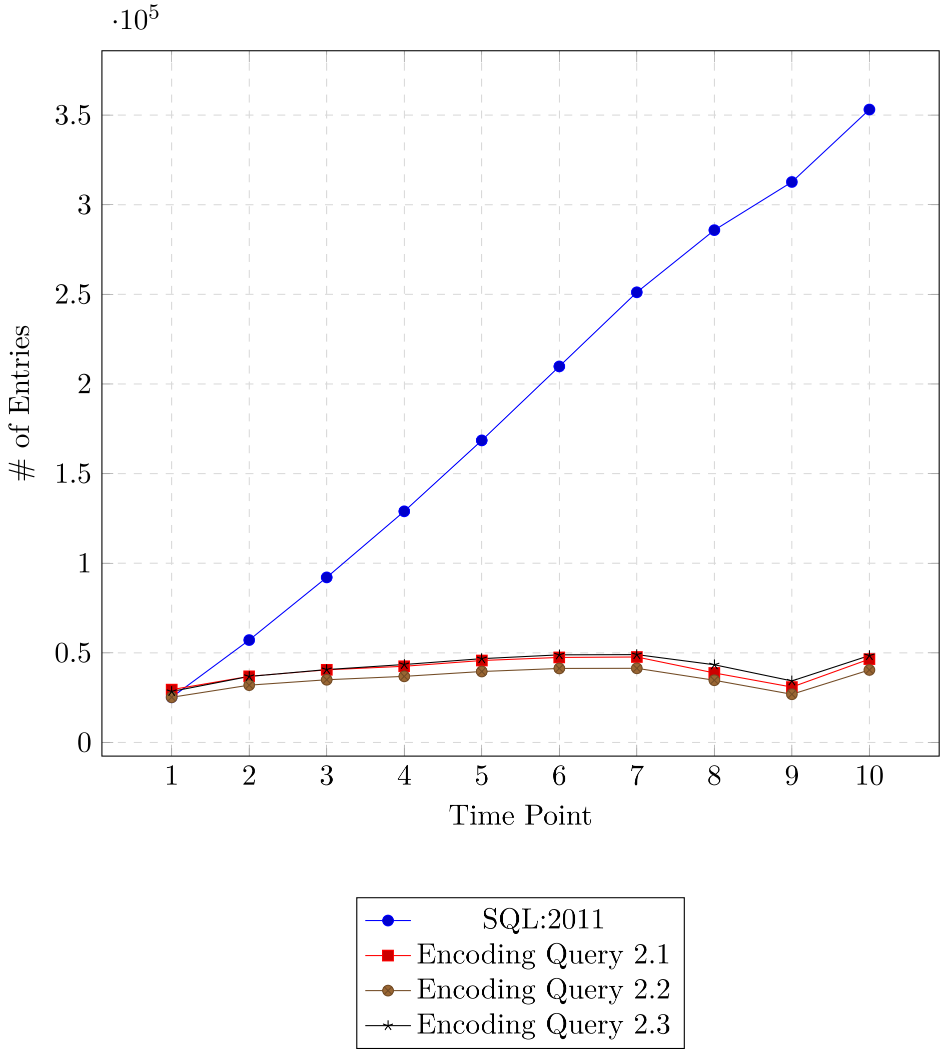

Figure 4.1: Encoding size of practical temporal queries compared to complete history size

4.2 Evaluation of Random Temporal Queries

Up to a size of four operators, the BHE [1] behaves similarly for RTQs as for PTQs. For 90% of the queries, the BHE [1] stores fewer entries than the SVTT [11] approach and takes on average 375 milliseconds per query per time point. Starting at a size of five, the number of queries for which the BHE [1] stores fewer entries than the SVTT [11] approach falls rapidly. For five and six operators it is 75%, for seven and eight operators it is 45% and for nine and 10 operators it is 35%.

However, the evaluation of the RTQs over the daily collected data, a larger number of time points, shows that for all sizes of RTQs the number of queries for which the BHE [1] stores fewer entries than the SVTT [11] approach increases. Thus, whether the BHE [1] stores fewer entries than the SVTT [11] approach does not directly depend on the size of the query, but rather on the number of time points.

A deeper analysis of the implementation of the BHE [4] shows that only the answers to subqueries under past operators are stored. Let $\mathsf{PSub}(\phi)$ denote the subset of queries from $\mathsf{Sub}(\phi)$ that start with a past operator, analogously to $\mathsf{FSub}(\phi)$. With $\mathsf{PSub}(\phi)$ and the size boundary $|\mathsf{Sub}(\phi)| * 2^{|\mathsf{FSub}(\phi)|} * |\Delta^{\mathsf{N_V}}|$ from [1] an upper bound can be found for when the BHE [1] will store fewer entries than the SVTT [11] approach.

Theorem 4.2.1. *There is a time point t, such that*

$$t * |\Delta^{\mathsf{N_V}}| > \left(\sum_{\psi \in \mathsf{PSub}(\phi)} 2^{|\mathsf{FSub}(\psi)|} * |\Delta^{\mathsf{N_V}}| \right) + 1 * |\Delta^{\mathsf{N_V}}|.$$

Proof. To proof Theorem 4.2.1, it is first shown that the sum is an upper bound for the size of the encoding.

Lemma 4.2.2. *The size of the encoding is bounded by*

$$\sum_{\psi \in \mathsf{PSub}(\phi)} 2^{|\mathsf{FSub}(\psi)|} * |\Delta^{\mathsf{N_V}}|.$$

Proof. From [1] it is known that the size of the encoding is bounded by

$$|\mathsf{Sub}(\phi)| * 2^{|\mathsf{FSub}(\phi)|} * |\Delta^{\mathsf{N_V}}|.$$

As the analysis of the implementation of the BHE [4] showed, only answers to subqueries under past operators are stored. Thus, $|\mathsf{Sub}(\phi)|$ can be substituted by $|\mathsf{PSub}(\phi)|$, resulting in a boundary of

$$|\mathsf{PSub}(\phi)| * 2^{|\mathsf{FSub}(\phi)|} * |\Delta^{\mathsf{N_V}}|.$$

This can be rewritten as

$$\sum_{\psi \in \mathsf{PSub}(\phi)} 2^{|\mathsf{FSub}(\phi)|} * |\Delta^{\mathsf{N_V}}|.$$

$|\mathsf{FSub}(\phi)|$ can be replaced by $|\mathsf{FSub}(\psi)|$, as for each subquery under a past operator which answers are stored, at most all subsets of Var_i^{ψ} need to be considered. This results in a boundary of

$$\sum_{\psi \in \mathsf{PSub}(\phi)} 2^{|\mathsf{FSub}(\psi)|} * |\Delta^{\mathsf{N_V}}|.$$

$\square$

This size boundary for the encoding can be used now to find t. The summand $1 * |\Delta^{\mathsf{N_V}}|$ indicates that additionally to the size of the encoding all entries of the current time point are stored. For the SVTT [11] approach, the size of the history for each time point i, $0 \leq i \leq n$ is

$$\sum_{0}^{i} |\Delta^{\mathsf{N_V}}| = i * |\Delta^{\mathsf{N_V}}|,$$

as at each time point, all entries are stored from then on. To find a time point t, from that the BHE [1] will store fewer entries than the SVTT [11] approach, find i, such that

$$i * |\Delta^{\mathsf{N_V}}| = \left(\sum_{\psi \in \mathsf{PSub}(\phi)} 2^{|\mathsf{FSub}(\psi)|} * |\Delta^{\mathsf{N_V}}| \right) + 1 * |\Delta^{\mathsf{N_V}}|$$

and then set $t = i + 1$. $\square$

Since $|\Delta^{\mathsf{N_V}}|$ can be omitted the data does not influence t, if assumed that the number of stored entries is the same at each time point. This proofs the assumption from Chapter 1, that it is irrelevant which exact brands and models are collected.

The size boundary presented here is smaller than or equal to the one from [1], therefore allowing a more precise estimate of when the BHE [1] will be advantageous.

Proposition 4.2.3.

$$\sum_{\psi \in \mathsf{PSub}(\phi)} 2^{|\mathsf{FSub}(\psi)|} * |\Delta^{\mathsf{N_V}}| \leq |\mathsf{Sub}(\phi)| * 2^{|\mathsf{FSub}(\phi)|} * |\Delta^{\mathsf{N_V}}|$$

Proof.

$$|\mathsf{FSub}(\psi)| \leq |\mathsf{FSub}(\phi)|$$

If $|\mathsf{FSub}(\psi)| = |\mathsf{FSub}(\phi)|$ for all $\psi \in \mathsf{PSub}(\phi)$ in order for the boundaries to be the same size, $|\mathsf{PSub}(\phi)| = |\mathsf{Sub}(\phi)|$ needs to hold and $|\mathsf{FSub}(\psi)| = |\mathsf{FSub}(\phi)| = 0$. Thus, only in the case that ϕ contains no future operators, the boundaries are the same size, else the one presented here is smaller. □

For every $t \leq 10$ and every query up to t the BHE [1] stores fewer entries than the SVTT [11] approach. However, t is an upper bound, and therefore even for queries for which t is extremely high the BHE [1] can be advantageous in even a few time points. For 43% of the RTQs with $t > 10$, the BHE [1] stores fewer entries than the SVTT [11] approach.

The average time to answer one query per time point also increases with t as can be seen in Table 4.1.

t	time in milliseconds
$t <= 10$	297
$10 < t <= 100$	3,379
$100 < t <= 1,000$	4,270
$1,000 < t <= 10,000$	10,727
$t > 10,000$	47,002

Table 4.1: Average time to answer a query

The size of the encodings for Query 41 ($t = 8$), Query 36 ($t = 15$), and Query 98 ($t = 33,554,690$) compared to the complete history can be seen in Figure 4.2.

The evaluation of the RTQs over a partial period and the daily collected data provides appropriate results.

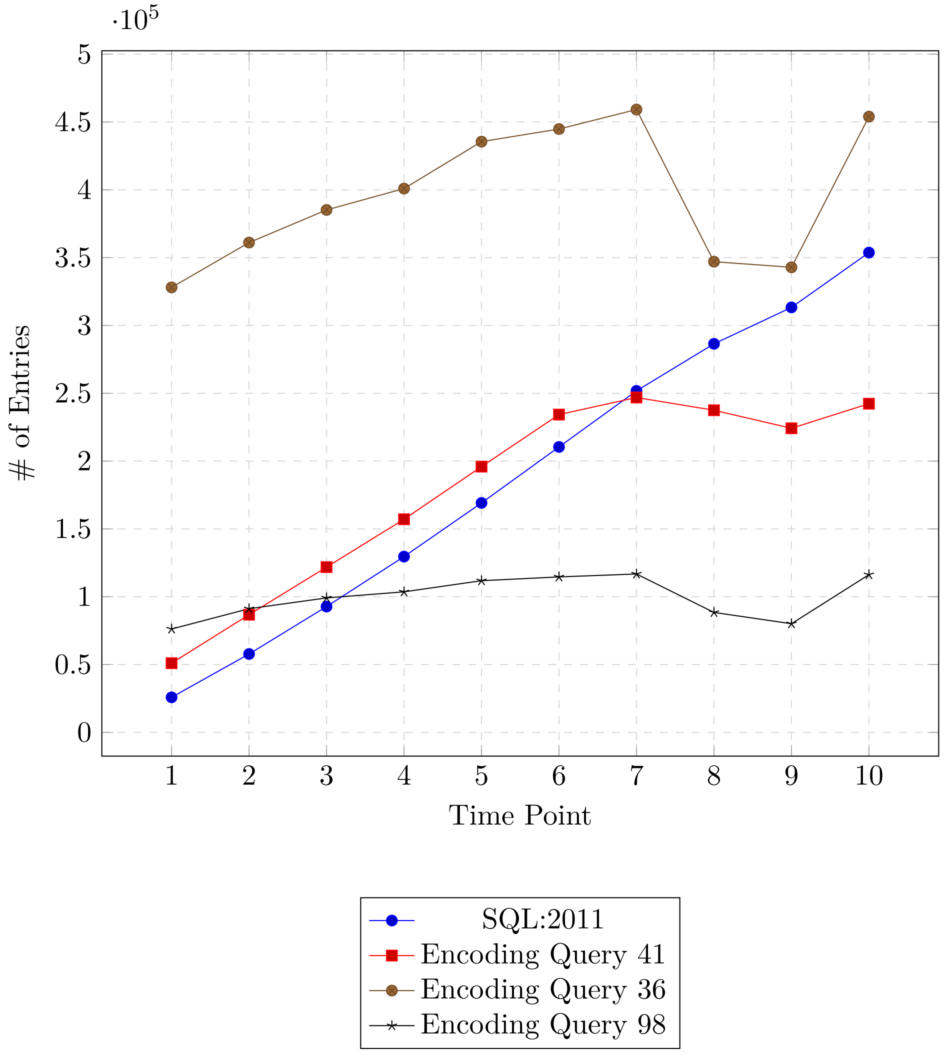

Figure 4.2: Encoding size of random temporal queries compared to complete history size

4.3 Discussion

As the evaluation in 4.2 has shown, a constant number of entries at any given time point is beneficial to estimate from which time point on the BHE [1] is smaller than the SVTT [11] approach. Thus, a constant data stream would give more accurate results than the variable data stream provided by the scraper [6].

Additionally, the time available was very limited. An investigation over a longer period would be able to confirm the trends observed. Since the evaluation of the queries was done by hand, it was not possible to evaluate significantly more queries, which would possibly allow further conclusions.

The evaluation in 4.2 also showed that the size of the queries, defined by the number of operators, has no direct influence on the results. Therefore, the size of the queries could be better defined using the number of subqueries or the nesting depth of future operators and past operators. This would possibly emphasize the difference between the boundary found in this thesis and the one mentioned in [1].

To be able to express the PTQs in the language of [1], the algorithm from [1] had to be extended. The extension by a filter operator could not be developed generically for all relational query languages. Thus, the PTQs can be answered on an SQL database, but for other database systems, a different solution has to be found.

Conclusion

In this thesis, the BHE from [1] was investigated to find out, how helpful it is in a practical application. Criteria to assess the degree of helpfulness are the usefulness of the answers, the time it takes to answer one query per time point, and the size of the encoding. The practical application in this thesis was an observation of an online automotive marketplace. The investigation was performed based on PTQs and RTQs.

The PTQs were defined to determine how the BHE [1] performs for meaningful queries. To be able to evaluate the PTQs, the BHE [1] had to be extended. As the evaluation showed, the algorithm can provide relevant answers at each time point within a short time. The number of stored entries was always significantly lower than with the SVTT [11] approach. From this, it follows that the algorithm is helpful to answer PTQs.

However, the PTQs were not diverse enough to cover the whole language from [1]. As a result, RTQs were defined to investigate further how queries influence the time to answer one query per time point and the size of the encoding. The evaluation showed that the size of the queries, defined by the number of operators, had no direct influence on neither the time to answer on query per time point nor the size of the encoding. Further investigation of the implementation [4] showed that there is an upper bound for the size of the encoding, which can be used to estimate from which time point t on the implementation [4] provides an encoding which is smaller than the SVTT [11] approach. This was confirmed by the evaluation, since for all queries for which $t \leq 10$, from t on, the encoding was smaller than the SVTT [11] approach. The boundary found in this thesis is smaller than the one mentioned in [1].

Outlook

In future work, the BHE [1] can be evaluated over a longer period of time on a constant data stream. This should make the advantages of the algorithm even more obvious.

Furthermore, MTOs could be adjusted so that $\mathsf{Sub}(\phi)$ corresponds to the intuitive meaning, i.e. $\mathsf{Sub}(\Diamond_6(\psi)) = \{\psi\}$. For this, a solution is needed, how to manage the 6 time points, which $\Diamond_6(\psi)$ needs, in one table.

Finally, a comparison of the queries, executed on other approaches, would be insightful to better assess the time required by the BHE [1] to answer the queries.

Bibliography

[1] Stefan Borgwardt, Marcel Lippmann, and Veronika Thost. Temporalizing rewritable query languages over knowledge bases. *Journal of Web Semantics*, 33:50–70, 2015.

[2] Jan Chomicki. Efficient checking of temporal integrity constraints using bounded history encoding. *ACM Transactions on Database Systems (TODS)*, 20(2):149–186, 1995.

[3] Joshua Schneider, David Basin, Srđan Krstić, and Dmitriy Traytel. A formally verified monitor for metric first-order temporal logic. In *International Conference on Runtime Verification*, pages 310–328. Springer, 2019.

[4] Thure Nebendahl. Bounded History Encoding Implementation.

[5] AutoScout24 GmbH.

[6] Thure Nebendahl. AutoScout24 Scraper.

[7] Christopher Buhtz. AutoScout24 Mining (Teil 1) - Webscraping mit Python.

[8] Leonard Richardson. Beautiful soup documentation. April 2007.

[9] Jeff Reback, Wes McKinney, jbrockmendel, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, gfyoung, Sinhrks, Adam Klein, Matthew Roeschke, Simon Hawkins, Jeff Tratner, Chang She, William Ayd, Terji Petersen, Marc Garcia, Jeremy Schendel, Andy Hayden, MomIsBestFriend, Vytautas Jancauskas, Pietro Battiston, Skipper Seabold, chris b1, h vetinari, Stephan Hoyer, Wouter Overmeire, alimcmaster1, Kaiqi Dong, Christopher Whelan, and Mortada Mehyar. pandas-dev/pandas: Pandas 1.0.3, mar 2020.

[10] Michael Bayer. SQLAlchemy. In Amy Brown and Greg Wilson, editors, *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks*. aosabook.org, 2012.

[11] Krishna Kulkarni and Jan-Eike Michels. Temporal features in SQL: 2011. *ACM Sigmod Record*, 41(3):34–43, 2012.

Appendix A

Complete Data Specifications

A.1 Complete list of all models

- AC: Cobra
- Alfa Romeo: 145, 146, 147, 155, 156, 159, 164, 4c, 75, 8c, 90, 6, Alfasud, Alfetta, Brera, Giulia, Gt, Gtv, Mito, Montreal, Puadrifoglio, Rz, Spider, Sprint, Sz
- Alpina: B3, B4, B5, B6, B7, B8, B9, B10, B11, B12, C1, C2, D10, D3, D4, D5, RoadsterS
- Aston Martin: Cygnet, DB, DB7, DB9, DB11, DBS, DBX, Lagonda, Rapide, V8, Valkyrie, Vantage, Virage, Volante
- Bentley: Arnage, Azure, Bentayga, Brooklands, Continental, Eight, FlyingSpur, Mulsanne, S1, S2, S3, Turbo R, Turbo RT, Turbo S,
- Bmw: 1er (all), 2002, 2er (all), 3er (all), 4er (all), 5er (all), 6er (all), 7er (all), 8er (all), M-Reihe (all), M1, Z-Reihe (all),
- Bugatti: Centodieci, Chiron, Divo, EB 110, EB 112, Veyron
- DeTomaso: all
- Ferrari: 195, 206, 208, 246, 250, 275, 288, 308, 328, 330, 348, 360, 365, 400, 412, 430Scuderia, 456, 458, 488, 512, 550, 575, 599, 612, 750, 812, California, Daytona, Dino GT4, Enzo Ferrari, F12, F355, F40, F430, F50, F512, F8 Spider, F8 Tributo, FF, FXX, GTC4 Lusso, LaFerrari, Mondial, Monza, Portofino, Roma, Scuderia Spider 16M, SF90 Stradale, Superamerica, Testarossa
- Fiat: 124 Spider, 500, 500 Abarth, Panda

- Honda: NSX
- Jaguar: 420, D-Type, Daimler, E-Pace, E-Type, F-Pace, F-Type, I-Pace, MK II, S-Type, X-Type, X300, XE, XF, XJ, XJ12, XJ40, XJ6, XJ8, XJR, XJS, XJSC, XK, XK8, XKR
- Lamborghini: Asterion, Aventador, Centenario, Countach, Diablo, Espada, Estoque, Gallardo, Huracan, Jalpa, Lm, Miura, Murciélago, Reventon, SianFkp 37, TerzoMillennio, UrracoP250, Urus, Veneno
- Lancia: all
- Land Rover: Defender, Discovery, Discovery Sport, Range Rover, Range Rover Evoque, Range Rover Sport, Range Rover Velar
- Lotus: all
- Maserati: all
- Mazda: RX-7, RX-8
- McLaren: all
- Mercedes-Benz: 190, C-Klasse (all), CL (all), CLK (all), CLS (all), E-Klasse (all), G-Klasse (all), S-Klasse (all), SL (all), SLC (all), SLK (all), SLR, SLS
- MG: all
- Mini: 1000, 1300, Cooper, Cooper S, John Cooper Works, Cooper Cabrio, Cooper S Cabrio, John Cooper Works Cabrio , Cooper Clubman, Cooper S Clubman, John Cooper Works Clubman
- Nissan: GT-R, Skyline
- Porsche: 365, 550, 718 Spider, 911, 930, 964, 991, 992, 993, 996, 997, 912, 918, 924, 928, 944, 959, 968, Boxter, Carrera GT, Panamera, Taycan
- Renault: Alpine A110, alpine A310, R5
- Rolls-Royce: all
- Ruf: all
- Tvr: all

A.2 Complete list of all aspects

“ABS”, “Abstandstempomat”, “Airbag hinten”, “Alarmanlage”, “Alufelgen”, “Anhängerkupplung”, “Anzahl Türen”, “Armlehne”, “Außenfarbe”, “Ausstattung”, “Beheizbare Frontscheibe”, “Beheizbares Lenkrad”, “Behindertengerecht”, “Beifahrerairbag”, “Berganfahrassistent”, “Bluetooth”, “Bordcomputer”, “CD”, “DAB-Radio”, “Dachreling”, “Einparkhilfe Kamera”, “Einparkhilfe selbstlenkendes System”, “Einparkhilfe Sensoren hinten”, “Einparkhilfe Sensoren vorne”, “Elektr. Fensterheber”, “Elektrische Heckklappe”, “Elektrische Seitenspiegel”, “Elektrische Sitze”, “Erstzulassung”, “ESP”, “Fahrerairbag”, “Fahrzeughalter”, “Farbe laut Hersteller”, “Feinstaubplakette”, “Freisprecheinrichtung”, “Garantie”, “Getönte”, “Getriebe”, “Getriebeart”, “Händler”, “Head-up display”, “HU Prüfung”, “HU/AU neu”, “Hubraum”, “Innenausstattung”, “Isofix”, “Karosserieform”, “Katalysator”, “Kilometerstand”, “Klimaanlage”, “Klimaautomatik”, “Kopfairbag”, “Kraftstoff”, “Kraftstoffverbrauch”, “Kurvenlicht”, “Land”, “LED-Scheinwerfer”, “LED-Tagfahrlicht”, “Lederlenkrad”, “Leistung”, “Lichtsensor”, “Lordosenstütze”, “Luftfederung”, “Marke”, “Massagesitze”, “Modell”, “MP3”, “Müdigkeitswarnsystem”, “Multifunktionslenkrad”, “Nachtsicht-Assistent”, “Navigationssystem”, “Nebelscheinwerfer”, “Nichtraucherfahrzeug”, “Notbremsassistent”, “Notrufsystem”, “Ort”, “Panoramadach”, “Preis”, “Radio”, “Rechtslenker”, “Regensensor”, “Reifendruckkontrollsystem”, “Schadstoffklasse”, “Schaltwippen”, “Scheckheftgepflegt”, “Scheiben”, “Schiebedach”, “Schiebetür”, “Schlüssellose Zentralverriegelung”, “Seitenairbag”, “Servolenkung”, “Sitzbelüftung”, “Sitzheizung”, “Sitzplätze”, “Skisack”, “Soundsystem”, “Sportfahrwerk”, “Sportpaket”, “Sportsitze”, “Sprachsteuerung”, “Spurhalteassistent”, “Standheizung”, “Start/Stop-Automatik”, “Tagfahrlicht”, “teilb. Rücksitzbank”, “Tempomat”, “Totwinkel-Assistent”, “Touchscreen”, “Traktionskontrolle”, “Tuning”, “TV”, “USB”, “Verkehrszeichenerkennung”, “Wegfahrsperre”, “Windschott(für Cabrio)”, “Winterreifen”, “Xenonscheinwerfer”, “Zentralverriegelung”, “Zentralverriegelung mit Funkfernbedienung”

Appendix B

Complete Queries

B.1 Complete list of Practical Temporal Queries

These queries are given in the language from [1] and in Java-Syntax.

1. Which cars price increased some time in the past?

   - $\diamond^-$(SELECT * FROM (SELECT * FROM ϕ) WHERE $p1 < p2$ [$\bigcirc^-$(SELECT *url*, *price* AS $p1$ FROM *autos*) $\cap$ SELECT *url*, *price* AS $p2$ FROM *autos*])
   - Query query1 = new EventuallyPast(new Filter (“SELECT * FROM phi WHERE p1 < p2”,new Conjunction(new Strong-Previous(new AtemporalQuery (“SELECT url, price AS p1 FROM autos”)), new AtemporalQuery (“SELECT url, price AS p2 FROM autos”))));

2. Which cars price decreased by more than 20% some time in the past?

   - $\diamond^-$(SELECT * FROM (SELECT * FROM ϕ) WHERE $0.8 * p1 > p2$[$\bigcirc^-$(SELECT *url*, *price* AS $p1$ FROM *autos*) $\cap$ SELECT *url*, *price* AS $p2$ FROM *autos*])
   - Query query1 = new EventuallyPast (new Filter(“SELECT * FROM phi WHERE 0.8*p1 > p2”,new Conjunction(new Strong-Previous(new AtemporalQuery (“SELECT url, price AS p1 FROM autos”)), new AtemporalQuery (“SELECT url, price AS p2 FROM autos”))));

3. For which cars did the attributes “url”, “price” and “kilometerstand” never change?

   - $\square^-$($\bullet^-$(SELECT *url*, *price*, *kilometerstand* FROM *autos*) $\cap$ SELECT *url*, *price*, *kilometerstand* FROM *autos*)

- Query query1 = new AlwaysPast(new Conjunction(new WeakPrevious(new AtemporalQuery (“SELECT url, price, kilometerstand FROM autos”)), new AtemporalQuery (“SELECT url, price, kilometerstand FROM autos”)));

4. Which cars price decreased by more than 20% from one time point to the next some time in the past 4 time points?

   - $\Diamond^{-}_{4}$(SELECT * FROM (SELECT * FROM ϕ) WHERE $0.8*p1 > p2$[$\bigcirc^{-}$(SELECT *url*, *price* AS $p1$ FROM *autos*) $\cap$ SELECT *url*, *price* AS $p2$ FROM *autos*])
   - Query query1 = new EventuallyPastPredicate(new Filter(“SELECT * FROM phi WHERE 0.8*p1 > p2”,new Conjunction(new StrongPrevious(new AtemporalQuery (“SELECT url, price AS p1 FROM autos”)), new AtemporalQuery (“SELECT url, price AS p2 FROM autos”))), 4);

5. Which brands have an increased average price compared to the last point in time?

   - SELECT *marke* FROM (SELECT * FROM ϕ) WHERE $p < q$[$\bigcirc^{-}$(SELECT *marke*, AVG(*price*) AS p FROM *autos* GROUP BY *marke*) $\cap$ SELECT *marke*, AVG(*price*) AS q FROM *autos* GROUP BY *marke*]
   - Query query1 = new Filter (“SELECT marke FROM phi WHERE p < q”,new Conjunction(new StrongPrevious(new AtemporalQuery (“SELECT marke, AVG(price) AS p FROM autos GROUP BY marke”)), new AtemporalQuery (“SELECT marke, AVG(price) AS q FROM autos GROUP BY marke”)));

6. Which models have an increased average price compared to the last point in time?

   - SELECT *modell* FROM (SELECT * FROM ϕ) WHERE $p < q$[$\bigcirc^{-}$(SELECT *modell*, AVG(*price*) AS p FROM *autos* GROUP BY *modell*) $\cap$ SELECT *modell*, AVG(*price*) AS q FROM *autos* GROUP BY *modell*]
   - Query query1 = new Filter(“SELECT modell FROM phi WHERE p < q”,new Conjunction(new StrongPrevious(new AtemporalQuery (“SELECT modell, AVG(price) AS p FROM autos GROUP BY modell”)), new AtemporalQuery (“SELECT

modell, AVG(price) AS q FROM autos GROUP BY modell"))
);

7. For which brands did the average price never drop from one time point to the next in the last 5 time points?

   - $\Box_5^-$(SELECT *marke* FROM (SELECT * FROM ϕ) WHERE $p <= q$)[$\bigcirc^-$(SELECT *marke*, AVG(*price*) AS *p* FROM *autos* GROUP BY *marke*) $\cap$ SELECT *marke*, AVG(*price*) AS *q* FROM *autos* GROUP BY *marke*]
   - Query query1 = new AlwaysPastPredicate(new Filter("SELECT marke FROM phi WHERE p <= q", new Conjunction (new StrongPrevious (new AtemporalQuery ("SELECT marke , AVG(price) AS p FROM autos GROUP BY marke")), new AtemporalQuery ("SELECT marke, AVG(price) AS q FROM autos GROUP BY marke"))), 5);

8. For which models did the average price never drop from one time point to the next in the last 5 time points?

   - $\Box_5^-$(SELECT *modell* FROM (SELECT * FROM ϕ) WHERE $p <= q$)[$\bigcirc^-$(SELECT *modell*, AVG(*price*) AS *p* FROM *autos* GROUP BY *modell*) $\cap$ SELECT *modell*, AVG(*price*) AS *q* FROM *autos* GROUP BY *modell*]
   - Query query1 = new AlwaysPastPredicate(new Filter("SELECT modell FROM phi WHERE p <= q", new Conjunction(new StrongPrevious (new AtemporalQuery ("SELECT modell, AVG(price) AS p FROM autos GROUP BY modell")), new AtemporalQuery ("SELECT modell, AVG(price) AS q FROM autos GROUP BY modell"))), 5);

9. For which brands did the amount of offers decrease?

   - SELECT *marke* FROM (SELECT * FROM ϕ) WHERE $p > q$[$\bigcirc^-$(SELECT *marke*, COUNT(*url*) AS *p* FROM *autos* GROUP BY *marke*) $\cap$ SELECT *marke*, COUNT(*url*) AS *q* FROM *autos* GROUP BY *marke*]
   - Query query1 =new Filter("SELECT marke FROM phi WHERE p > q",new Conjunction(new StrongPrevious(new AtemporalQuery ("SELECT marke, COUNT(url) AS p FROM autos GROUP BY marke")), new AtemporalQuery ("SELECT marke, COUNT(url) AS q FROM autos GROUP BY marke")));

10. For which brands did the average price drop by more than 10% some time in the past?

    - $\Diamond^-$(SELECT *marke* FROM (SELECT * FROM ϕ) WHERE $0.9 * p > q[\bigcirc^-$(SELECT *marke*, AVG(*price*) AS *p* FROM *autos* GROUP BY *marke*) $\cap$ SELECT *marke*, AVG(*price*) AS *q* FROM *autos* GROUP BY *marke*])
    - Query query1 = new EventuallyPast(new Filter("SELECT marke FROM phi WHERE 0.9*p > q",new Conjunction(new StrongPrevious(new AtemporalQuery ("SELECT marke, AVG(price) AS p FROM autos GROUP BY marke")), new AtemporalQuery ("SELECT marke, AVG(price) AS q FROM autos GROUP BY marke"))));

11. Which cars were always present until the time point before they were sold?

    - $\bigcirc^-(\Box^-$(SELECT *url* FROM *autos*)) $\cap$ SELECT *url* FROM *autos* WHERE *deleted*
    - Query query1 = new Conjunction(new StrongPrevious(new AlwaysPast(new AtemporalQuery ("SELECT url FROM autos"))), new AtemporalQuery ("SELECT url FROM autos WHERE deleted"));

12. Which cars cost less than 10,000 at the last point in time and cost more than 10,000 at this point in time?

    - $\bigcirc^-$(SELECT *url* FROM *autos* WHERE *price* < 10.000) $\cap$ SELECT *url* FROM *autos* WHERE *price* > 10.000
    - Query query1 = new Conjunction(new StrongPrevious(new AtemporalQuery ("SELECT url FROM autos WHERE price < 10.000")), new AtemporalQuery ("SELECT url FROM autos WHERE price > 10.000"));

13. Which cars cost more than 1.000.000 some time in the last 6 points in time?

    - $\Diamond^-_6$(SELECT *url* FROM *autos* WHERE *price* > 1000000)
    - Query query1 = new EventuallyPastPredicate(new AtemporalQuery ("SELECT url FROM autos WHERE price > 1000000"), 6);

14. Which cars had more than 100.00km at the last time point and were sold at this time point?

    - $\bigcirc^-$(SELECT *url* FROM *autos* WHERE *kilometerstand* $>$ 100000) $\cap$ SELECT *url* FROM *autos* WHERE *deleted*
    - Query query1 = new Conjunction(new StrongPrevious(new AtemporalQuery (“SELECT url FROM autos WHERE kilometerstand > 100000”)), new AtemporalQuery (“SELECT url FROM autos WHERE deleted”));

15. Which cars have cost more than 1.000.000 since it cost less than 1.000.000 until it was sold?

    - $\bigcirc^-$(SELECT *url* FROM *autos* WHERE *price* $>$ 1000000 S SELECT *url* FROM *autos* WHERE *price* $<$ 1000000) $\cap$ SELECT *url* FROM *autos* WHERE *deleted*
    - Query query1 = new Conjunction (new StrongPrevious (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE price > 1000000”), new AtemporalQuery (“SELECT url FROM autos WHERE price < 1000000”))), new AtemporalQuery (“SELECT url FROM autos WHERE deleted”));

Query	*i*	$\lvert BHE\rvert/\lvert History\rvert$
1	3	0.24
2	3	0.23
3	3	0.24
4	6	0.23
5	2	0.11
6	2	0.12
7	7	0.11
8	7	0.11
9	2	0.11
10	3	0.11
11	3	0.14
12	2	0.13
13	7	0.14
14	2	0.13
15	3	0.22

B.2 Complete list of Random Temporal Queries

These queries are given in Java-Syntax as the algorithm to generate these queries returns queries in this syntax.

1. new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 6);

2. new WeakNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”));

3. new AlwaysPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3);

4. new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”));

5. new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”));

6. new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9);

7. new Eventually (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”));

8. new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”),new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”));

9. new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”));

10. new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 7);

11. new StrongNext (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)));

12. new AlwaysPredicate (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 7);

13. new Disjunction (new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”));

14. new WeakPreviousPredicate (new AlwaysPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 0), 6);

15. new WeakPrevious (new Conjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)));

16. new WeakNextPredicate (new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 5);

17. new Conjunction (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”));

18. new StrongPreviousPredicate (new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 0);

19. new WeakPrevious (new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 0));

20. new EventuallyPastPredicate (new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 0);

21. new EventuallyPredicate (new Eventually (new AlwaysPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 2)), 3);

22. new StrongPrevious (new EventuallyPredicate (new EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 0), 2));

23. new Conjunction (new AlwaysPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 1), new WeakPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 6));

24. new SincePredicate (new AlwaysPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 8), new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 8);

25. new AlwaysPastPredicate (new WeakNextPredicate (new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 2), 7);

26. new WeakPreviousPredicate (new WeakNext (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 3);

27. new Eventually (new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))));

28. new StrongPrevious (new WeakNextPredicate (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 0));

29. new StrongPrevious (new StrongPreviousPredicate (new WeakPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9), 9));

30. new SincePredicate (new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3), 0);

31. new AlwaysPast (new Always (new StrongPrevious (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)))));

32. new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AlwaysPredicate (new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 1)));

33. new StrongPrevious (new AlwaysPastPredicate (new WeakNextPredicate (new Conjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 9), 4));

34. new Eventually (new Until (new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new Eventually (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))));

35. new Conjunction (new Conjunction (new UntilPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 6), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new StrongNextPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 8));

36. new EventuallyPredicate (new UntilPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new SincePredicate (new AlwaysPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 4), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9), 5), 9);

37. new Conjunction (new Disjunction (new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)));

38. new AlwaysPast (new Eventually (new StrongNextPredicate (new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 0)));

39. new StrongNext (new EventuallyPastPredicate (new StrongNext-Predicate (new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 1), 2));

40. new StrongPrevious (new StrongNext (new UntilPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 7), 5)));

41. new Disjunction (new UntilPredicate (new AtemporalQuery (“SE-LECT url FROM autos WHERE NOT deleted”) , new StrongNext-Predicate (new WeakPreviousPredicate (new AtemporalQuery (“SE-LECT url FROM autos WHERE NOT deleted”), 6), 2), 2), new Eventually (new AtemporalQuery (“SELECT url FROM au-tos WHERE NOT deleted”)));

42. new StrongNext (new WeakPreviousPredicate (new StrongNext (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 5)), 9));

43. new StrongNext (new WeakPrevious (new WeakNextPredicate (new UntilPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new WeakNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 4), 4)));

44. new Disjunction (new Always (new Conjunction (new Atemporal-Query (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), new AlwaysPastPredicate (new WeakNext (new Atemporal-Query (“SELECT url FROM autos WHERE NOT deleted”)), 4));

45. new StrongNext (new SincePredicate (new AtemporalQuery (“SE-LECT url FROM autos WHERE NOT deleted”), new Always (new StrongPrevious (new WeakPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 6))), 3));

46. new SincePredicate (new SincePredicate (new StrongPrevious (new EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3)), new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 2), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 5);

47. new WeakPrevious (new WeakNextPredicate (new AlwaysPast (new EventuallyPastPredicate (new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 5)), 4));

48. new WeakPreviousPredicate (new WeakPrevious (new StrongNextPredicate (new Since (new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 4)), 3);

49. new WeakNext (new EventuallyPast (new AlwaysPastPredicate (new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 2)));

50. new StrongPrevious (new StrongNextPredicate (new StrongNext (new StrongNextPredicate (new EventuallyPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 9)), 2));

51. new UntilPredicate (new EventuallyPastPredicate (new AlwaysPredicate (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 7), 3), new Conjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new WeakPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 1);

52. new StrongPreviousPredicate (new Conjunction (new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 7), new EventuallyPredicate (new Disjunction (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE

NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 7)), 9);

53. new WeakPreviousPredicate (new StrongNextPredicate (new Always (new SincePredicate (new AlwaysPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 4), new AlwaysPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9), 2)), 5), 0);

54. new StrongNextPredicate (new EventuallyPastPredicate (new EventuallyPredicate (new Since (new WeakNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 4), 3), 5);

55. new Disjunction (new StrongNext (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 4)), new StrongNextPredicate (new AlwaysPredicate (new Conjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 3), 7));

56. new StrongNextPredicate (new WeakPrevious (new StrongNextPredicate (new Until (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 6), new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 3)), 9);

57. new StrongPrevious (new Eventually (new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new WeakPrevious (new WeakNextPredicate (new EventuallyPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 6)))));

58. new WeakPreviousPredicate (new StrongPrevious (new Until (new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 1), new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”),

new Eventually (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))))), 1);

59. new StrongPreviousPredicate (new Conjunction (new EventuallyPastPredicate (new AlwaysPredicate (new WeakPrevious (new EventuallyPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3)), 5), 7), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 6);

60. new WeakPreviousPredicate (new AlwaysPastPredicate (new StrongNextPredicate (new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new WeakPreviousPredicate (new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 4)), 4), 0), 4);

61. new UntilPredicate (new AlwaysPastPredicate (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Eventually (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 4), new Eventually (new StrongPrevious (new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)))), 3);

62. new Conjunction (new AlwaysPastPredicate (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 6), new AlwaysPredicate (new StrongPreviousPredicate (new StrongPrevious (new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 9), 5));

63. new EventuallyPredicate (new StrongNextPredicate (new WeakPrevious (new EventuallyPredicate (new StrongPrevious (new Eventually (new StrongNextPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3))), 4)), 4), 2);

64. new SincePredicate (new UntilPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 0), 6), new AlwaysPast (new StrongNextPredicate (new AlwaysPastPredicate (new AlwaysPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9), 4), 4)), 8);

65. new StrongPrevious (new StrongNextPredicate (new AlwaysPast (new EventuallyPastPredicate (new EventuallyPredicate (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 8), 1), 7)), 1));

66. new Until (new AlwaysPredicate (new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 6), new EventuallyPredicate (new StrongPrevious (new WeakNext (new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)))), 5));

67. new StrongNext (new AlwaysPast (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AlwaysPast (new AlwaysPastPredicate (new WeakPreviousPredicate (new WeakPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 9), 2)))));

68. new AlwaysPastPredicate (new AlwaysPredicate (new WeakNextPredicate (new StrongPrevious (new WeakPrevious (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))))), 2), 7), 8);

69. new Conjunction (new WeakNextPredicate (new WeakNextPredicate (new StrongNextPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 4), 2), 8), new StrongNext (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new WeakNextPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 2))));

70. new AlwaysPredicate (new UntilPredicate (new Eventually (new EventuallyPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), new StrongPrevious (new Eventually (new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 4))), 0), 7);

71. new AlwaysPast (new Until (new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”

), 8), new StrongNext (new EventuallyPredicate (new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new WeakNext (new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)))), 2))));

72. new StrongPrevious (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Until (new Conjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Disjunction (new StrongPrevious (new AlwaysPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9)), new EventuallyPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)))), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))));

73. new StrongPrevious (new EventuallyPastPredicate (new WeakPrevious (new WeakNext (new AlwaysPastPredicate (new AlwaysPredicate (new Always (new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 1), 3))), 8));

74. new AlwaysPast (new EventuallyPastPredicate (new WeakNextPredicate (new AlwaysPastPredicate (new Always (new AlwaysPredicate (new StrongPreviousPredicate (new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 6), 8)), 8), 7), 0));

75. new StrongPrevious (new Until (new StrongNext (new EventuallyPast (new StrongNextPredicate (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 4))), new EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 5)));

76. new StrongNextPredicate (new SincePredicate (new Disjunction (new WeakPrevious (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), new StrongNext (new AlwaysPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 2))), new

EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 5), 4), 6);

77. new EventuallyPastPredicate (new Conjunction (new StrongNextPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 5), new Eventually (new Eventually (new SincePredicate (new StrongPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 1)))), 5);

78. new WeakNextPredicate (new Always (new AlwaysPredicate (new WeakNextPredicate (new UntilPredicate (new AlwaysPast (new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), new WeakNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 6), 3), 5)), 3);

79. new StrongPrevious (new EventuallyPredicate (new WeakPrevious (new StrongPrevious (new WeakPrevious (new Disjunction (new Until (new WeakPrevious (new AtemporalQuery("SELECT url FROM autos WHERE NOT deleted")), new AtemporalQuery("SELECT url FROM autos WHERE NOT deleted")), new AtemporalQuery("SELECT url FROM autos WHERE NOT deleted")))))), 2));

80. new WeakPreviousPredicate (new Since (new SincePredicate (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new WeakNextPredicate (new EventuallyPastPredicate (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 7), 5)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9), new WeakPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 3);

81. new WeakNext (new Conjunction (new StrongPrevious (new StrongPreviousPredicate (new Conjunction (new EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 8), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 4)), new Since (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT

deleted”), 2), new EventuallyPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 0)))));

82. new AlwaysPredicate (new AlwaysPast (new Eventually (new AlwaysPastPredicate (new Eventually (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Always (new Eventually (new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)))), 3)), 6))), 9);

83. new Always (new Always (new Since (new Always (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 5), 9)), new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Always (new EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 4)), 7))));

84. new AlwaysPastPredicate (new Eventually (new WeakNextPredicate (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new EventuallyPredicate (new WeakPreviousPredicate (new SincePredicate (new EventuallyPredicate (new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3), 5), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 6), 5), 3), 7), 4)), 8);

85. new AlwaysPastPredicate (new EventuallyPast (new WeakNextPredicate (new UntilPredicate (new Conjunction (new WeakNextPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 2), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new StrongPreviousPredicate (new AlwaysPastPredicate (new Conjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 2), 1), 8), 5)), 5);

86. new Eventually (new AlwaysPastPredicate (new UntilPredicate (new StrongNextPredicate (new EventuallyPastPredicate (new StrongNextPredicate (new WeakPreviousPredicate (new Eventually (new

WeakPrevious (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 0), 8), 8), 7), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3), 1));

87. new Conjunction (new WeakPreviousPredicate (new StrongPrevious (new StrongPreviousPredicate (new Eventually (new WeakPrevious-Predicate (new WeakNextPredicate (new Conjunction (new Atem-poralQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 4), 7)), 1)), 1), new StrongPrevious (new Atem-poralQuery (“SELECT url FROM autos WHERE NOT deleted”)));

88. new StrongNextPredicate (new StrongPreviousPredicate (new Since (new SincePredicate (new Until (new AlwaysPredicate (new Atem-poralQuery (“SELECT url FROM autos WHERE NOT deleted”), 1), new WeakNextPredicate (new WeakPreviousPredicate (new Atem-poralQuery (“SELECT url FROM autos WHERE NOT deleted”), 7), 6)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 8), new SincePredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Atempo-ralQuery (“SELECT url FROM autos WHERE NOT deleted”), 4)), 4), 2);

89. new EventuallyPredicate (new StrongNext (new Conjunction (new EventuallyPastPredicate (new WeakNextPredicate (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AlwaysPastPredicate (new WeakPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 8), 8)), 4), 4), new Conjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Atempo-ralQuery (“SELECT url FROM autos WHERE NOT deleted”)))), 5);

90. new Disjunction (new WeakNextPredicate (new Eventually (new Disjunction (new Until (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Atemporal-Query (“SELECT url FROM autos WHERE NOT deleted”)), new WeakPreviousPredicate (new StrongPreviousPredicate (new Atem-poralQuery (“SELECT url FROM autos WHERE NOT deleted”), 1), 3)), new AtemporalQuery (“SELECT url FROM autos WHERE

NOT deleted”))), 1), new EventuallyPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3));

91. new AlwaysPredicate (new Since (new Always (new EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9)), new StrongPreviousPredicate (new AlwaysPast (new AlwaysPredicate (new AlwaysPredicate (new WeakPreviousPredicate (new AlwaysPast (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 1), 8), 0)), 1)), 2);

92. new AlwaysPast (new UntilPredicate (new SincePredicate (new WeakNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AlwaysPastPredicate (new StrongPreviousPredicate (new UntilPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new EventuallyPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 5), 1), 3), 2), 5), new UntilPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new WeakPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 8), 3), 6));

93. new Conjunction (new AlwaysPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 5), new StrongPreviousPredicate (new WeakPrevious (new AlwaysPredicate (new Always (new StrongNext (new StrongPrevious (new Until (new Disjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)))))), 5)), 1));

94. new StrongNext (new WeakPreviousPredicate (new WeakNextPredicate (new StrongPrevious (new StrongPreviousPredicate (new StrongPreviousPredicate (new StrongPrevious (new StrongNext (new EventuallyPastPredicate (new Until (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 3)))), 7), 1)), 8), 7));

95. new WeakNext (new StrongPrevious (new EventuallyPastPredicate (new EventuallyPast (new EventuallyPredicate (new Disjunction (new EventuallyPredicate (new AtemporalQuery (“SELECT url

FROM autos WHERE NOT deleted”), 0), new EventuallyPast-Predicate (new Always (new Conjunction (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), 3)), 2)), 6)));

96. new Until (new Eventually (new Since (new StrongNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))), new SincePredicate (new WeakPrevious (new Weak-Next (new UntilPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new EventuallyPastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 5), 7))), new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 3));

97. new EventuallyPastPredicate (new Conjunction (new Eventually-PastPredicate (new WeakNext (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 1), new AlwaysPredicate (new AlwaysPast (new AlwaysPast (new SincePredicate (new Always-PastPredicate (new UntilPredicate (new AtemporalQuery (“SE-LECT url FROM autos WHERE NOT deleted”), new Atemporal-Query (“SELECT url FROM autos WHERE NOT deleted”), 3), 0), new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 1))), 1)), 1);

98. new StrongPreviousPredicate (new WeakPrevious (new UntilPredicate (new UntilPredicate (new Disjunction (new EventuallyPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 4), new StrongNextPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 3)), new EventuallyPastPredicate (new StrongNextPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 7), 2), 6), new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”)), 7)), 0);

99. new AlwaysPast (new StrongNext (new StrongNext (new Conjunction (new Always (new Since (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), new Always (new Weak-Next (new AlwaysPastPredicate (new AtemporalQuery (“SELECT

url FROM autos WHERE NOT deleted”), 8))))), new Always (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”))))));

100. new WeakNext (new Disjunction (new AlwaysPast (new Eventually (new WeakNext (new EventuallyPastPredicate (new Eventually-PastPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 8), 7))))), new Eventually (new EventuallyPredicate (new StrongPreviousPredicate (new AtemporalQuery (“SELECT url FROM autos WHERE NOT deleted”), 9), 3)))));

$Query$	i	$\lvert BHE\rvert / \lvert History\rvert$ weekly	$\lvert BHE\rvert / \lvert History\rvert$ daily
1	7	0.76	0.34
2	1	0.11	$4.94 \cdot 10^{-2}$
3	1	0.11	$4.94 \cdot 10^{-2}$
4	1	0.11	$4.94 \cdot 10^{-2}$
5	2	0.22	$9.79 \cdot 10^{-2}$
6	10	1.08	0.49
7	1	0.11	$4.94 \cdot 10^{-2}$
8	1	0.11	$4.94 \cdot 10^{-2}$
9	1	0.11	$4.94 \cdot 10^{-2}$
10	8	0.78	0.38
11	1	0.11	$4.94 \cdot 10^{-2}$
12	1	0.11	$4.94 \cdot 10^{-2}$
13	1	0.11	$4.94 \cdot 10^{-2}$
14	7	0.68	0.33
15	2	0.22	$9.79 \cdot 10^{-2}$
16	2	0.22	$9.79 \cdot 10^{-2}$
17	1	0.11	$4.94 \cdot 10^{-2}$
18	2	0.22	$9.79 \cdot 10^{-2}$
19	2	0.22	$9.79 \cdot 10^{-2}$
20	2	0.22	$9.79 \cdot 10^{-2}$
21	3	0.29	0.14
22	5	0.22	$9.79 \cdot 10^{-2}$
23	7	0.68	0.33
24	2,050	1.05	0.66
25	57	0.43	0.23
26	7	0.3	0.15
27	2	0.22	$9.79 \cdot 10^{-2}$
28	3	0.22	$9.79 \cdot 10^{-2}$
29	20	1.02	0.85
30	5	0.4	0.21

$Query$	i	$\|BHE\|/\|History\|$ weekly	$\|BHE\|/\|History\|$ daily
31	5	0.22	$9.79 \cdot 10^{-2}$
32	5	0.28	0.14
33	2,561	0.11	$4.94 \cdot 10^{-2}$
34	1	0.11	$4.94 \cdot 10^{-2}$
35	1	0.11	$4.94 \cdot 10^{-2}$
36	14	1.28	0.62
37	2	0.14	$6.63 \cdot 10^{-2}$
38	4	0.34	0.16
39	9	0.33	0.15
40	72	0.86	0.39
41	7	0.68	0.33
42	47	1.16	0.62
43	513	0.11	$4.94 \cdot 10^{-2}$
44	9	0.36	0.19
45	14	1.05	0.51
46	66	1.92	1.03
47	24	0.31	0.15
48	67	0.34	0.15
49	8	0.78	0.39
50	4,098	0.34	0.13
51	386	0.75	0.38
52	1,161	1.24	1.03
53	1,029	0.7	0.36
54	197	0.76	0.34
55	5	0.54	0.24
56	24	0.78	0.36
57	322	0.45	0.18
58	12	0.74	0.48
59	421	5.02	2.5
60	265	0.43	0.23

Query	i	$\lvert BHE\rvert/\lvert History\rvert$ weekly	$\lvert BHE\rvert/\lvert History\rvert$ daily
61	11	0.73	0.4
62	28	1.1	0.71
63	273	0.11	$4.94 \cdot 10^{-2}$
64	8,222	7.58	4.98
65	29	2.13	0.86
66	4	0.22	$9.79 \cdot 10^{-2}$
67	15	1.02	0.63
68	8,199	0.74	0.42
69	5	0.28	0.14
70	8	0.81	0.37
71	137	3.83	2.61
72	14	1.13	0.63
73	185	1.27	0.83
74	69,640	0.99	0.67
75	4,131	0.72	0.32
76	260	1.87	0.97
77	644	0.72	0.38
78	3	0.17	$8.33 \cdot 10^{-2}$
79	15	0.88	0.47
80	425	14.83	9.33
81	1,284	1.37	0.67
82	78	0.61	0.51
83	303	8.92	5.1
84	67,684	20.06	11.86
85	$1.97 \cdot 10^{5}$	91.34	390
86	$5.28 \cdot 10^{5}$	0.33	0.15
87	210	0.85	0.39
88	3,340	34.72	17
89	81	1.37	0.7
90	8	0.71	0.34

$Query$	i	$\vert BHE\vert/\vert History\vert$ weekly	$\vert BHE\vert/\vert History\vert$ daily
91	$2.63 \cdot 10^5$	10.51	47
92	2,092	14.79	200
93	520	0.84	0.46
94	7,215	2.52	1.17
95	71	2.8	1.1
96	1,033	1.54	0.74
97	59	1.05	0.64
98	$3.36 \cdot 10^7$	0.33	0.15
99	77	0.65	0.4
100	29	3.66	1.42

Appendix C

Complete Proofs

Proposition 3.1.2 (ct. Proposition 3.4 in [1]). *For* $\mathfrak{a} : \mathsf{FVar}(\phi) \to \mathsf{N_C}$ *and* $0 \leq i \leq n$,

1. $\mathfrak{I}, i \models \mathfrak{a}(\Box\phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *and*
   - $i < n$ *implies* $\mathfrak{I}, i+1 \models \mathfrak{a}(\Box\phi_1)$
2. $\mathfrak{I}, i \models \mathfrak{a}(\Box^-\phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *and*
   - $i > 0$ *implies* $\mathfrak{I}, i-1 \models \mathfrak{a}(\Box^-\phi_1)$
3. $\mathfrak{I}, i \models \mathfrak{a}(\Diamond\phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or*
   - $i < n$ *and* $\mathfrak{I}, i+1 \models \mathfrak{a}(\Diamond\phi_1)$
4. $\mathfrak{I}, i \models \mathfrak{a}(\Diamond^-\phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or*
   - $i > 0$ *and* $\mathfrak{I}, i-1 \models \mathfrak{a}(\Diamond^-\phi_1)$

Proof. The proof works mainly on the basis of semantics.

1. $\Box\phi_1 \equiv \phi_1 \wedge \CIRCLE\Box\phi_1$

$$
\begin{aligned}
& \mathfrak{I}, i \models \mathfrak{a}(\Box\phi_1) && \text{(C.1)}\\
\Leftrightarrow & \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, i \leq k \leq n && \text{(C.2)}\\
\Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i < n \text{ implies} && \text{(C.3)}\\
& \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, i+1 \leq k \leq n) &&\\
\Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i < n \text{ implies } \mathfrak{I}, i+1 \models \mathfrak{a}(\Box\phi_1)) && \text{(C.4)}\\
\Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1 \wedge \CIRCLE\Box\phi_1) && \text{(C.5)}
\end{aligned}
$$

(C.3) is equivalent to (C.2) because

- in case $i < n$, the query needs to be satisfied now, at time point i, and at all future time points k, $i+1 \leq k \leq n$, to be satisfied. Since $i < n$ is true, the satisfaction of future time points depends solely on the second part of the "implies"-statement; and
- in case $i = n$, the query needs to be satisfied now, at time point i, to be satisfied. There are no future time points k, $n+1 \leq k \leq n$. Since $i = n$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $n \leq k \leq n$, and $i < n$ is not true, thus the "implies"-statement does not affect satisfaction.

2. $\Box^- \phi_1 \equiv \phi_1 \wedge \bullet^- \Box^- \phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\Box^- \phi_1) \tag{C.6}$$
$$\Leftrightarrow \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, 0 \leq k \leq i \tag{C.7}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i > 0 \text{ implies} \tag{C.8}$$
$$\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, 0 \leq k \leq i-1)$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i > 0 \text{ implies } \mathfrak{I}, i-1 \models \mathfrak{a}(\Box^- \phi_1)) \tag{C.9}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1 \wedge \bullet^- \Box^- \phi_1) \tag{C.10}$$

(C.8) is equivalent to (C.7) because

- in case $i > 0$, the query needs to be satisfied now, at time point i, and at all past time points k, $0 \leq k \leq i-1$, to be satisfied. Since $i > 0$ is true, the satisfaction of past time points depends solely on the second part of the "implies"-statement; and
- in case $i = 0$, the query needs to be satisfied now, at time point i, to be satisfied. There are no past time points k, $0 \leq k \leq 0-1$. Since $i = 0$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $0 \leq k \leq 0$, and $i > 0$ is not true, thus the "implies"-statement does not affect satisfaction.

3. $\Diamond \phi_1 \equiv \phi_1 \vee \bigcirc \Diamond \phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\Diamond \phi_1) \tag{C.11}$$
$$\Leftrightarrow \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, i \leq k \leq n \tag{C.12}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i < n \text{ and} \tag{C.13}$$
$$\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, i+1 \leq k \leq n)$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i < n \text{ and } \mathfrak{I}, i+1 \models \mathfrak{a}(\Diamond \phi_1)) \tag{C.14}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1 \vee \bigcirc \Diamond \phi_1) \tag{C.15}$$

(C.13) is equivalent to (C.12) because

- in case $i < n$, the query needs to be satisfied now, at time point i, or at any future time point k, $i+1 \leq k \leq n$, to be satisfied. Since $i < n$ is true, the satisfaction of future time points depends solely on the second part of the "and"-statement; and
- in case $i = n$, the query needs to be satisfied now, at time point i, to be satisfied. There are no future time points k, $n+1 \leq k \leq n$. Since $i = n$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $n \leq k \leq n$, and $i > 0$ is not true, thus the "and"-statement does not affect satisfaction.

4. $\Diamond^- \phi_1 \equiv \phi_1 \vee \bigcirc^- \Diamond^- \phi_1$

$$
\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\Diamond^- \phi_1) && \text{(C.16)}\\
\Leftrightarrow &\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, 0 \leq k \leq i && \text{(C.17)}\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i > 0 \text{ and} && \text{(C.18)}\\
&\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, 0 \leq k \leq i-1) &&\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i > 0 \text{ and } \mathfrak{I}, i-1 \models \mathfrak{a}(\Diamond^- \phi_1)) && \text{(C.19)}\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \vee \bigcirc^- \Diamond^- \phi_1) && \text{(C.20)}
\end{aligned}
$$

(C.18) is equivalent to (C.17) because

- in case $i > 0$, the query needs to be satisfied now, at time point i, or at any past time point k, $0 \leq k \leq i-1$, to be satisfied. Since $i > 0$ is true, the satisfaction of past time points depends solely on the second part of the "and"-statement; and
- in case $i = 0$, the query needs to be satisfied now, at time point i, to be satisfied. There are no past time points k, $0 \leq k \leq 0-1$. Since $i = 0$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $0 \leq k \leq 0$, and $i > 0$ is not true, thus the "and"-statement does not affect satisfaction.

□

Lemma 3.1.6 (ct. Lemma 6.4 in [1])**.** *If Φ_{i-1} for $\Box, \Box^-, \Diamond$ and $\Diamond^-$ is correct for i-1, then Φ_i^0 for $\Box, \Box^-, \Diamond$ and $\Diamond^-$ is correct for i.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_i^0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$ for all $n \geq i$.

If $\psi = \Box^{-}\psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(\Phi_{i-1}(\psi)) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i-1) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \Diamond^{-}\psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cup \mathsf{eval}^n(\Phi_{i-1}(\psi)) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cup \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i-1) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \Box\psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(x_i^{\psi}) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i \\ \Delta^{\mathsf{N_V}} & \text{if } n = i \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \Diamond\psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cup \mathsf{eval}^n(x_i^{\psi}) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cup \left\{ \begin{array}{ll} \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i \\ \emptyset & \text{if } n = i \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

□

Proposition 3.3.2 (ct. Proposition 3.4 in [1])**.** *For* $\mathfrak{a} : \mathsf{FVar}(\phi) \to \mathsf{N_C}$, $0 \le i \le n$ *and* $p = 0$, $\mathfrak{I}, i \models \mathfrak{a}(\phi)$ *iff* $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or* $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$, *respectively.*

Proof. The proof works mainly on the basis of semantics.

1. $\bigcirc_0\phi_1 \equiv \phi_1$

$$\begin{aligned}&\mathfrak{I}, i \models \mathfrak{a}(\bigcirc_0\phi_1) && \text{(C.21)} \\ \Leftrightarrow\, & i + 0 \le n \text{ and } \mathfrak{I}, i + 0 \models \mathfrak{a}(\phi_1) && \text{(C.22)} \\ \Leftrightarrow\, & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) && \text{(C.23)}\end{aligned}$$

2. $\CIRCLE_0 \phi_1 \equiv \phi_1$

$$\begin{aligned} & \mathfrak{I}, i \models \mathfrak{a}(\CIRCLE_0 \phi_1) && \text{(C.24)} \\ \Leftrightarrow & i + 0 \leq n \text{ implies } \mathfrak{I}, i + 0 \models \mathfrak{a}(\phi_1) && \text{(C.25)} \\ \Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) && \text{(C.26)} \end{aligned}$$

3. $\bigcirc_0^- \phi_1 \equiv \phi_1$

$$\begin{aligned} & \mathfrak{I}, i \models \mathfrak{a}(\bigcirc_0^- \phi_1) && \text{(C.27)} \\ \Leftrightarrow & i - 0 \geq 0 \text{ and } \mathfrak{I}, i - 0 \models \mathfrak{a}(\phi_1) && \text{(C.28)} \\ \Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) && \text{(C.29)} \end{aligned}$$

4. $\CIRCLE_0^- \phi_1 \equiv \phi_1$

$$\begin{aligned} & \mathfrak{I}, i \models \mathfrak{a}(\CIRCLE_0^- \phi_1) && \text{(C.30)} \\ \Leftrightarrow & i - 0 \geq 0 \text{ implies } \mathfrak{I}, i - 0 \models \mathfrak{a}(\phi_1) && \text{(C.31)} \\ \Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) && \text{(C.32)} \end{aligned}$$

5. $\Box_0 \phi_1 \equiv \phi_1$

$$\begin{aligned} & \mathfrak{I}, i \models \mathfrak{a}(\Box_0 \phi_1) && \text{(C.33)} \\ \Leftrightarrow & \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, i \leq k \leq \mathsf{min}(i + 0, n) && \text{(C.34)} \\ \Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) && \text{(C.35)} \end{aligned}$$

6. $\Box_0^- \phi_1 \equiv \phi_1$

$$\begin{aligned} & \mathfrak{I}, i \models \mathfrak{a}(\Box_0^- \phi_1) && \text{(C.36)} \\ \Leftrightarrow & \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, \mathsf{max}(i - 0, 0) \leq k \leq i && \text{(C.37)} \\ \Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) && \text{(C.38)} \end{aligned}$$

7. $\Diamond_0 \phi_1 \equiv \phi_1$

$$\begin{aligned} & \mathfrak{I}, i \models \mathfrak{a}(\Diamond_0 \phi_1) && \text{(C.39)} \\ \Leftrightarrow & \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, i \leq k \leq \mathsf{min}(i + 0, n) && \text{(C.40)} \\ \Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) && \text{(C.41)} \end{aligned}$$

8. $\Diamond_0^- \phi_1 \equiv \phi_1$

$$\begin{aligned} & \mathfrak{I}, i \models \mathfrak{a}(\Diamond_0^- \phi_1) && \text{(C.42)} \\ \Leftrightarrow & \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, \mathsf{max}(i - 0, 0) \leq k \leq i && \text{(C.43)} \\ \Leftrightarrow & \mathfrak{I}, i \models \mathfrak{a}(\phi_1) && \text{(C.44)} \end{aligned}$$

9. $\phi_1 \mathsf{U}_0 \phi_2 \equiv \phi_2$

$$\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{U}_0 \phi_2) \tag{C.45}$$

$$\Leftrightarrow \text{there is } k, i \leq k \leq \mathsf{min}(i+0, n), \text{ with } \mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and} \tag{C.46}$$
$$\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, i \leq j < k$$

$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) \tag{C.47}$$

10. $\phi_1 \mathsf{S}_0 \phi_2 \equiv \phi_2$

$$\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{S}_0 \phi_2) \tag{C.48}$$

$$\Leftrightarrow \text{there is } k, \mathsf{max}(i-0, 0) \leq k \leq i, \text{ with } \mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and} \tag{C.49}$$
$$\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, k < j \leq i$$

$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) \tag{C.50}$$

□

Proposition 3.3.3 (ct. Proposition 3.4 in [1]). *For* $\mathfrak{a} : \mathsf{FVar}(\phi) \to \mathsf{N_C}$, $0 \leq i \leq n$ *and* $p > 0$,

1. $\mathfrak{I}, i \models \mathfrak{a}(\bigcirc_p \phi_1)$ *iff*
   - $i < n$ *and* $\mathfrak{I}, i+1 \models \mathfrak{a}(\bigcirc_{p-1} \phi_1)$
2. $\mathfrak{I}, i \models \mathfrak{a}(\bullet_p \phi_1)$ *iff*
   - $i < n$ *implies* $\mathfrak{I}, i+1 \models \mathfrak{a}(\bullet_{p-1} \phi_1)$
3. $\mathfrak{I}, i \models \mathfrak{a}(\bigcirc^-_p \phi_1)$ *iff*
   - $i > 0$ *and* $\mathfrak{I}, i-1 \models \mathfrak{a}(\bigcirc^-_{p-1} \phi_1)$
4. $\mathfrak{I}, i \models \mathfrak{a}(\bullet^-_p \phi_1)$ *iff*
   - $i > 0$ *implies* $\mathfrak{I}, i-1 \models \mathfrak{a}(\bullet^-_{p-1} \phi_1)$
5. $\mathfrak{I}, i \models \mathfrak{a}(\Box_p \phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *and*
   - $i < n$ *implies* $\mathfrak{I}, i+1 \models \mathfrak{a}(\Box_{p-1} \phi_1)$
6. $\mathfrak{I}, i \models \mathfrak{a}(\Box^-_p \phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *and*

- $i > 0$ *implies* $\mathfrak{I}, i-1 \models \mathfrak{a}(\Box^{-}_{p-1}\phi_1)$

7. $\mathfrak{I}, i \models \mathfrak{a}(\Diamond_p \phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or*
   - $i < n$ *and* $\mathfrak{I}, i+1 \models \mathfrak{a}(\Diamond_{p-1}\phi_1)$

8. $\mathfrak{I}, i \models \mathfrak{a}(\Diamond^{-}_p \phi_1)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ *or*
   - $i > 0$ *and* $\mathfrak{I}, i-1 \models \mathfrak{a}(\Diamond^{-}_{p-1}\phi_1)$

9. $\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{U}_p \phi_2)$ *iff*
   - $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$ *or*
   - $\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1)$ *and* $i < n$ *and* $\mathfrak{I}, i+1 \models \mathfrak{a}(\phi_1 \mathsf{U}_{p-1} \phi_2)$

10. $\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{S}_p \phi_2)$ *iff*
    - $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ *or*
    - $\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1)$ *and* $i > 0$ *and* $\mathfrak{I}, i-1 \models \mathfrak{a}(\phi_1 \mathsf{S}_{p-1} \phi_2)$

Proof. The proof works mainly on the basis of semantics.

1. $\bigcirc_p \phi_1 \equiv \bigcirc\bigcirc_{p-1}\phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\bigcirc_p \phi_1) \tag{C.51}$$
$$\Leftrightarrow i + p \leq n \text{ and } \mathfrak{I}, i+p \models \mathfrak{a}(\phi_1) \tag{C.52}$$
$$\Leftrightarrow i < n \text{ and } (i+1) + (p-1) \leq n \text{ and } \mathfrak{I}, (i+1)+(p-1) \models \mathfrak{a}(\phi_1) \tag{C.53}$$
$$\Leftrightarrow i < n \text{ and } \mathfrak{I}, i+1 \models \mathfrak{a}(\bigcirc_{p-1}\phi_1) \tag{C.54}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\bigcirc\bigcirc_{p-1}\phi_1) \tag{C.55}$$

2. $\bullet_p \phi_1$ is equivalent to $\bullet\bullet_{p-1}\phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\bullet_p \phi_1) \tag{C.56}$$
$$\Leftrightarrow i + p \leq n \text{ implies } \mathfrak{I}, i+p \models \mathfrak{a}(\phi_1) \tag{C.57}$$
$$\Leftrightarrow i < n \text{ implies } (i+1) + (p-1) \leq n \text{ implies } \mathfrak{I}, (i+1)+(p-1) \models \mathfrak{a}(\phi_1) \tag{C.58}$$
$$\Leftrightarrow i < n \text{ implies } \mathfrak{I}, i+1 \models \mathfrak{a}(\bullet_{p-1}\phi_1) \tag{C.59}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\bullet\bullet_{p-1}\phi_1) \tag{C.60}$$

3. $\bigcirc^-_p \phi_1$ is equivalent to $\bigcirc^- \bigcirc^-_{p-1}\phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\bigcirc^-_p \phi_1) \tag{C.61}$$
$$\Leftrightarrow i - p \geq 0 \text{ and } \mathfrak{I}, i - p \models \mathfrak{a}(\phi_1) \tag{C.62}$$
$$\Leftrightarrow i > 0 \text{ and } (i-1) - (p-1) \geq 0 \text{ and } \mathfrak{I}, (i-1) - (p-1) \models \mathfrak{a}(\phi_1) \tag{C.63}$$
$$\Leftrightarrow i > 0 \text{ and } \mathfrak{I}, i - 1 \models \mathfrak{a}(\bigcirc^-_{p-1}\phi_1) \tag{C.64}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\bigcirc^- \bigcirc^-_{p-1}\phi_1) \tag{C.65}$$

4. $\bullet^-_p \phi_1$ is equivalent to $\bullet^- \bullet^-_{p-1}\phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\bullet^-_p \phi_1) \tag{C.66}$$
$$\Leftrightarrow i - p \geq 0 \text{ implies } \mathfrak{I}, i - p \models \mathfrak{a}(\phi_1) \tag{C.67}$$
$$\Leftrightarrow i > 0 \text{ implies } (i-1) - (p-1) \geq 0 \text{ implies } \mathfrak{I}, (i-1) - (p-1) \models \mathfrak{a}(\phi_1) \tag{C.68}$$
$$\Leftrightarrow i > 0 \text{ implies } \mathfrak{I}, i - 1 \models \mathfrak{a}(\bullet^-_{p-1}\phi_1) \tag{C.69}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\bullet^- \bullet^-_{p-1}\phi_1) \tag{C.70}$$

5. $\Box_p\phi_1 \equiv \phi_1 \wedge \bullet\Box_{p-1}\phi_1$

$$\mathfrak{I}, i \models \mathfrak{a}(\Box_p\phi_1) \tag{C.71}$$
$$\Leftrightarrow \mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, i \leq k \leq \mathsf{min}(i+p, n) \tag{C.72}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i < n \text{ implies} \tag{C.73}$$
$$\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, i+1 \leq k \leq \mathsf{min}((i+1)+(p-1), n))$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i < n \text{ implies } \mathfrak{I}, i+1 \models \mathfrak{a}(\Box_{p-1}\phi_1)) \tag{C.74}$$
$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_1 \wedge \bullet\Box_{p-1}\phi_1) \tag{C.75}$$

(C.73) is equivalent to (C.72) because

- in case $i < n$, the query needs to be satisfied now, at time point i, and at future time points k, $i+1 \leq k \leq \mathsf{min}((i+1)+(p-1), n)$, to be satisfied. Since $i < n$ is true, the satisfaction of future time points depends solely on the second part of the "implies"-statement; and
- in case $i = n$, the query needs to be satisfied now, at time point i, to be satisfied. There are no future time points k, $n+1 \leq k \leq \mathsf{min}((i+1)+(p-1), n)$. Since $i = n$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is

equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $n \le k \le \mathsf{min}(i+p, n)$, and $i < n$ is not true, thus the "implies"-statement does not affect satisfaction.

6. $\Box^-_p \phi_1 \equiv \phi_1 \wedge \CIRCLE^- \Box^-_p \phi_1$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\Box^-_p \phi_1) && \text{(C.76)}\\
\Leftrightarrow &\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, \mathsf{max}(i-p, 0) \le k \le i && \text{(C.77)}\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i > 0 \text{ implies} && \text{(C.78)}\\
&\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for all } k, \mathsf{max}((i-1)-(p-1), 0) \le k \le i-1)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ and } (i > 0 \text{ implies } \mathfrak{I}, i-1 \models \mathfrak{a}(\Box^-_{p-1}\phi_1)) && \text{(C.79)}\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \wedge \CIRCLE^- \Box^-_{p-1}\phi_1) && \text{(C.80)}
\end{aligned}$$

(C.78) is equivalent to (C.77) because

- in case $i > 0$, the query needs to be satisfied now, at time point i, and at past time points k, $\mathsf{max}((i-1)-(p-1), 0) \le k \le i-1$, to be satisfied. Since $i > 0$ is true, the satisfaction of past time points depends solely on the second part of the "implies"-statement; and
- in case $i = 0$, the query needs to be satisfied now, at time point i, to be satisfied. There are no past time points k, $\mathsf{max}((i-1)-(p-1), 0) \le k \le 0-1$. Since $i = 0$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for all k, $\mathsf{max}(i-p, 0) \le k \le 0$, and $i > 0$ is not true, thus the "implies"-statement does not affect satisfaction.

7. $\Diamond_p \phi_1 \equiv \phi_1 \vee \bigcirc \Diamond_p \phi_1$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\Diamond_p \phi_1) && \text{(C.81)}\\
\Leftrightarrow &\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, i \le k \le \mathsf{min}(i+p, n) && \text{(C.82)}\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i < n \text{ and} && \text{(C.83)}\\
&\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, i+1 \le k \le \mathsf{min}((i+1)+(p-1), n))\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i < n \text{ and } \mathfrak{I}, i+1 \models \mathfrak{a}(\Diamond_{p-1}\phi_1)) && \text{(C.84)}\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \vee \bigcirc \Diamond_{p-1}\phi_1) && \text{(C.85)}
\end{aligned}$$

(C.83) is equivalent to (C.82) because

- in case $i < n$, the query needs to be satisfied now, at time point i, or at any of the future time points k, $i+1 \le k \le \mathsf{min}((i+1)+(p-1), n)$, to be satisfied. Since $i < n$ is true, the satisfaction of future time points depends solely on the second part of the "and"-statement; and

- in case $i = n$, the query needs to be satisfied now, at time point i, to be satisfied. There are no future time points k, $n + 1 \leq k \leq \mathsf{min}((i + 1) + (p - 1), n)$. Since $i = n$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $n \leq k \leq \mathsf{min}(i + p, n)$, and $i < n$ is not true, thus the "and"-statement does not affect satisfaction.

8. $\Diamond_p^- \phi_1 \equiv \phi_1 \vee \bigcirc^- \Diamond_p^- \phi_1$

$$\begin{aligned}
&\mathfrak{I}, i \models \mathfrak{a}(\Diamond_p^- \phi_1) && \text{(C.86)}\\
\Leftrightarrow &\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, \mathsf{max}(i - p, 0) \leq k \leq i && \text{(C.87)}\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i > 0 \text{ and} && \text{(C.88)}\\
&\mathfrak{I}, k \models \mathfrak{a}(\phi_1) \text{ for some } k, \mathsf{max}((i - 1) - (p - 1), 0) \leq k \leq i - 1)\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1) \text{ or } (i > 0 \text{ and } \mathfrak{I}, i - 1 \models \mathfrak{a}(\Diamond_{p-1}^- \phi_1)) && \text{(C.89)}\\
\Leftrightarrow &\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \vee \bigcirc^- \Diamond_{p-1}^- \phi_1) && \text{(C.90)}
\end{aligned}$$

(C.88) is equivalent to (C.87) because

- in case $i > 0$, the query needs to be satisfied now, at time point i, or at any of the past time points k, $\mathsf{max}((i - 1) - (p - 1), 0) \leq k \leq i - 1$, to be satisfied. Since $i > 0$ is true, the satisfaction of past time points depends solely on the second part of the "and"-statement; and
- in case $i = 0$, the query needs to be satisfied now, at time point i, to be satisfied. There are no past time points k, $\mathsf{max}((i-1)-(p-1), 0) \leq k \leq 0 - 1$. Since $i = 0$ is true, $\mathfrak{I}, i \models \mathfrak{a}(\phi_1)$ is equivalent to $\mathfrak{I}, k \models \mathfrak{a}(\phi_1)$ for some k, $\mathsf{max}(i - p, 0) \leq k \leq 0$, and $i > 0$ is not true, thus the "and"-statement does not affect satisfaction.

9. $\phi_1 \mathsf{U}_p \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge \bigcirc(\phi_1 \mathsf{U}_{p-1} \phi_2))$

$$\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{U}_p \phi_2) \tag{C.91}$$

$$\Leftrightarrow \text{there is } k, i \leq k \leq \mathsf{min}(i+p, n), \text{ with } \mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and } \mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, i \leq j < k \tag{C.92}$$

$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ or } (\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ and } (i < n \text{ and there is } k, i+1 \leq k \leq \mathsf{min}((i+1)+(p-1), n), \text{ with } \mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and } \mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, i+1 \leq j < k)) \tag{C.93}$$

$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ or } (\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ and } (i < n \text{ and } \mathfrak{I}, i+1 \models \mathfrak{a}(\phi_1 \mathsf{U}_{p-1} \phi_2))) \tag{C.94}$$

$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_2 \vee (\phi_1 \wedge \bigcirc(\phi_1 \mathsf{U}_{p-1} \phi_2))) \tag{C.95}$$

(C.93) is equivalent to (C.92) because

- in case $i < n$, either ϕ_2 needs to be satisfied now, at time point i, or ϕ_1 needs to be satisfied now, at time point i, and there needs to be a future time point k, $i+1 \leq k \leq \mathsf{min}((i+1)+(p-1), n)$, where ϕ_2 is satisfied and ϕ_1 is satisfied for all time points j, $i+1 \leq j < k$, for the query to be satisfied.
- in case $i = n$, ϕ_2 needs to be satisfied now, at time point i, for the query to be satisfied. There are no future time points k, $n+1 \leq k \leq \mathsf{min}((i+1)+(p-1), n)$. Since $i = n$ is true, $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$ is equivalent to there is k, $n \leq k \leq \mathsf{min}(i+p, n)$, with $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all $j, n \leq j < k$, and $i < n$ is not true, thus the "or"-statement does not affect satisfaction.

10. $\phi_1 \mathsf{S}_p \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge \bigcirc^-(\phi_1 \mathsf{S}_{p-1} \phi_2))$

$$\mathfrak{I}, i \models \mathfrak{a}(\phi_1 \mathsf{S}_p \phi_2) \tag{C.96}$$

$$\Leftrightarrow \text{there is } k, \mathsf{max}(i-p, 0) \leq k \leq i, \text{ with } \mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and } \mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, k < j \leq i \tag{C.97}$$

$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ or } (\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ and } (i > 0 \text{ and there is } k, \mathsf{max}((i-1)-(p-1), 0) \leq k \leq i-1, \text{ with } \mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ and } \mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ for all } j, k < j \leq i-1)) \tag{C.98}$$

$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2) \text{ or } (\mathfrak{I}, i \models \mathfrak{a}_{\phi_1}(\phi_1) \text{ and } (i > 0 \text{ and } \mathfrak{I}, i-1 \models \mathfrak{a}(\phi_1 \mathsf{S}_{p-1} \phi_2))) \tag{C.99}$$

$$\Leftrightarrow \mathfrak{I}, i \models \mathfrak{a}(\phi_2 \vee (\phi_1 \wedge \bigcirc^-(\phi_1 \mathsf{S}_{p-1} \phi_2))) \tag{C.100}$$

(C.98) is equivalent to (C.97) because

- in case $i > 0$, either ϕ_2 needs to be satisfied now, at time point i, or ϕ_1 needs to be satisfied now, at time point i, and there needs to be a past time point k, $\mathsf{max}((i-1)-(p-1),0) \leq k \leq i-1$, where ϕ_2 is satisfied and ϕ_1 is satisfied for all time points j, $k < j \leq i-1$, for the query to be satisfied.
- in case $i = 0$, ϕ_2 needs to be satisfied now, at time point i, for the query to be satisfied. There are no past time points k, $\mathsf{max}((i-1)-(p-1),0) \leq k \leq 0-1$. Since $i = 0$ is true, $\mathfrak{I}, i \models \mathfrak{a}_{\phi_2}(\phi_2)$ is equivalent to there is k, $\mathsf{max}(i-p,0) \leq k \leq 0$, with $\mathfrak{I}, k \models \mathfrak{a}_{\phi_2}(\phi_2)$ and $\mathfrak{I}, j \models \mathfrak{a}_{\phi_1}(\phi_1)$ for all $j, k < j \leq 0$, and $i > 0$ is not true, thus the "or"-statement does not affect satisfaction.

$\square$

Lemma 3.3.5 (ct. Lemma 6.3 in [1]). *The function Φ_0 with metric temporal operators is correct for 0.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$ for all $n \geq 0$. The missing cases can be found in Appendix C.
If $\psi = \bigcirc_0\psi_1$, $\psi = \bullet_0\psi_1$, $\psi = \bigcirc_0^-\psi_1$, $\psi = \bullet_0^-\psi_1$, $\psi = \Box_0\psi_1$, $\psi = \Box_0^-\psi_1$, $\psi = \Diamond_0\psi_1$ or $\psi = \Diamond_0^-\psi_1$, then

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{eval}^n(\Phi_0(\psi_1)).$$

This is by induction equal to $Ans(\psi_1, \mathfrak{I}^{(n)}, 0)$ which then is, as shown in Proposition 3.3.2, equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
If $\psi = \psi_1 \mathsf{U}_0 \psi_2$ or $\psi = \psi_1 \mathsf{S}_0 \psi_2$, then

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{eval}^n(\Phi_0(\psi_2)).$$

This is by induction equal to $Ans(\psi_2, \mathfrak{I}^{(n)}, 0)$ which then is, as shown in Proposition 3.3.2, equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.
If $\psi = \bigcirc_p^-\psi_1$, $\psi = \bullet_p^-\psi_1$, $\psi = \Box_p^-\psi_1$ or $\psi = \Diamond_p^-\psi_1$ and $p > 0$, then

$$\mathsf{eval}^n(\Phi_0(\psi)) = \mathsf{eval}^n(\Phi_0(\psi_1)).$$

This is by induction equal to $\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0)$ which then is, as shown in Proposition 3.3.3, equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)$.

If $\psi = \bigcirc_p \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(x_0^{\bigcirc_p \psi_1}) \\ &= \left\{ \begin{array}{ll} \mathsf{Ans}(\bigcirc_{p-1}\psi_1, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \emptyset & \text{if } n = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)\end{aligned}$$

If $\psi = \bullet_p \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(x_0^{\bullet_p \psi_1}) \\ &= \left\{ \begin{array}{ll} \mathsf{Ans}(\bullet_{p-1}\psi_1, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \Delta^{\mathsf{Nv}} & \text{if } n = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)\end{aligned}$$

If $\psi = \Box_p \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_1)) \cap \mathsf{eval}^n(x_0^{\Box_p \psi_1}) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\Box_{p-1}\psi_1, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \Delta^{\mathsf{Nv}} & \text{if } n = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)\end{aligned}$$

If $\psi = \Diamond_p \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_1)) \cup \mathsf{eval}^n(x_0^{\Diamond_p \psi_1}) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cup \left\{ \begin{array}{ll} \mathsf{Ans}(\Diamond_{p-1}\psi_1, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \emptyset & \text{if } n = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)\end{aligned}$$

If $\psi = \psi_1 \mathsf{U}_p \psi_2$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_2)) \cup (\mathsf{eval}^n(\Phi_0(\psi_1)) \cap \mathsf{eval}^n(x_0^{\psi_1 \mathsf{U}_p \psi_2})) \\ &= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, 0) \cup (\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, 0) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\psi_1 \mathsf{U}_{p-1}\psi_2, \mathfrak{I}^{(n)}, 1) & \text{if } n > 0 \\ \emptyset & \text{if } n = 0 \end{array} \right\}) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)\end{aligned}$$

If $\psi = \psi_1 \mathsf{S}_p \psi_2$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_0(\psi)) &= \mathsf{eval}^n(\Phi_0(\psi_2)) \\ &= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, 0) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, 0)\end{aligned}$$

$\square$

Lemma 3.3.6 (ct. Lemma 6.4 in [1]). *If Φ_{i-1} with metric temporal operators is correct for i-1, then Φ_i^0 with metric temporal operators is correct for i.*

Proof. It is shown by induction on the structure of the subqueries $\psi \in \mathsf{Sub}(\phi)$ that $\mathsf{eval}^n(\Phi_i^0(\psi))$ is equal to $\mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)$ for all $n \geq i$.

If $\psi = \bigcirc_0\psi_1$, $\psi = \bullet_0\psi_1$, $\psi = \bigcirc_0^-\psi_1$, $\psi = \bullet_0^-\psi_1$, $\psi = \Box_0\psi_1$, $\psi = \Box_0^-\psi_1$, $\psi = \Diamond_0\psi_1$ or $\psi = \Diamond_0^-\psi_1$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1))\\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i)\\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \psi_1 \mathsf{U}_0 \psi_2$ or $\psi = \psi_1 \mathsf{S}_0 \psi_2$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_2))\\ &= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, i)\\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \bigcirc_p\psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(x_i^{\bigcirc_p\psi_1})\\ &= \left\{\begin{array}{ll}\mathsf{Ans}(\bigcirc_{p-1}\psi_1, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i\\ \emptyset & \text{if } n = i\end{array}\right\}\\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \bullet_p\psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(x_i^{\bullet_p\psi_1})\\ &= \left\{\begin{array}{ll}\mathsf{Ans}(\bullet_{p-1}\psi_1, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i\\ \Delta^{\mathsf{N_V}} & \text{if } n = i\end{array}\right\}\\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \bigcirc_p^-\psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_{i-1}(\bigcirc_{p-1}^-\psi_1))\\ &= \left\{\begin{array}{ll}\mathsf{Ans}(\bigcirc_{p-1}^-\psi_1, \mathfrak{I}^{(n)}, i-1) & \text{if } i > 0\\ \emptyset & \text{if } i = 0\end{array}\right\}\\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \bullet_p^- \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_{i-1}(\bullet_{p-1}^- \psi_1)) \\ &= \left\{ \begin{array}{ll} \mathsf{Ans}(\bullet_{p-1}^- \psi_1, \mathfrak{I}^{(n)}, i-1) & \text{if } i > 0 \\ \Delta^{\mathsf{N_V}} & \text{if } i = 0 \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \Box_p \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(x_i^{\Box_p \psi_1}) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\Box_{p-1} \psi_1, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i \\ \Delta^{\mathsf{N_V}} & \text{if } n = i \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \Box_p^- \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(\Phi_{i-1}(\Box_p^- \psi_1)) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \mathsf{Ans}(\Box_{p-1}^- \psi_1, \mathfrak{I}^{(n)}, i-1) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \Diamond_p \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cup \mathsf{eval}^n(x_i^{\Diamond_{p-1} \psi_1}) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cup \left\{ \begin{array}{ll} \mathsf{Ans}(\Diamond_{p-1} \psi_1, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i \\ \emptyset & \text{if } n = i \end{array} \right\} \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \Diamond_p^- \psi_1$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_1)) \cup \mathsf{eval}^n(\Phi_{i-1}(\Box_{p-1}^- \psi_1)) \\ &= \mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cup \mathsf{Ans}(\Box_{p-1}^- \psi_1, \mathfrak{I}^{(n)}, i-1) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \psi_1 \mathsf{U}_p \psi_2$ and $p > 0$, then

$$\begin{aligned}\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_2)) \cup (\mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \mathsf{eval}^n(x_i^{\psi_1 \mathsf{U}_p \psi_2})) \\ &= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, i) \cup (\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \left\{ \begin{array}{ll} \mathsf{Ans}(\psi_1 \mathsf{U}_{p-1} \psi_2, \mathfrak{I}^{(n)}, i+1) & \text{if } n > i \\ \emptyset & \text{if } n = i \end{array} \right\}) \\ &= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)\end{aligned}$$

If $\psi = \psi_1 \mathsf{S}_p \psi_2$ and $p > 0$, then

$$\begin{aligned}
\mathsf{eval}^n(\Phi_i^0(\psi)) &= \mathsf{eval}^n(\Phi_i^0(\psi_2)) \cup (\mathsf{eval}^n(\Phi_i^0(\psi_1)) \cap \Phi_{i-1}(\psi_1 \mathsf{S}_{p-1} \psi_2)) \\
&= \mathsf{Ans}(\psi_2, \mathfrak{I}^{(n)}, i) \cup (\mathsf{Ans}(\psi_1, \mathfrak{I}^{(n)}, i) \cap \mathsf{Ans}(\psi_1 \mathsf{S}_{p-1} \psi_2, \mathfrak{I}^{(n)}, i-1)) \\
&= \mathsf{Ans}(\psi, \mathfrak{I}^{(n)}, i)
\end{aligned}$$

$\square$