

Offline Trace Checking of Quantitative Properties of Service-Based Applications

Srđan Krstić

with

Domenico Bianculli, Carlo Ghezzi and Pierluigi San Pietro



Offline Trace Checking of Quantitative Properties of Service-Based Applications

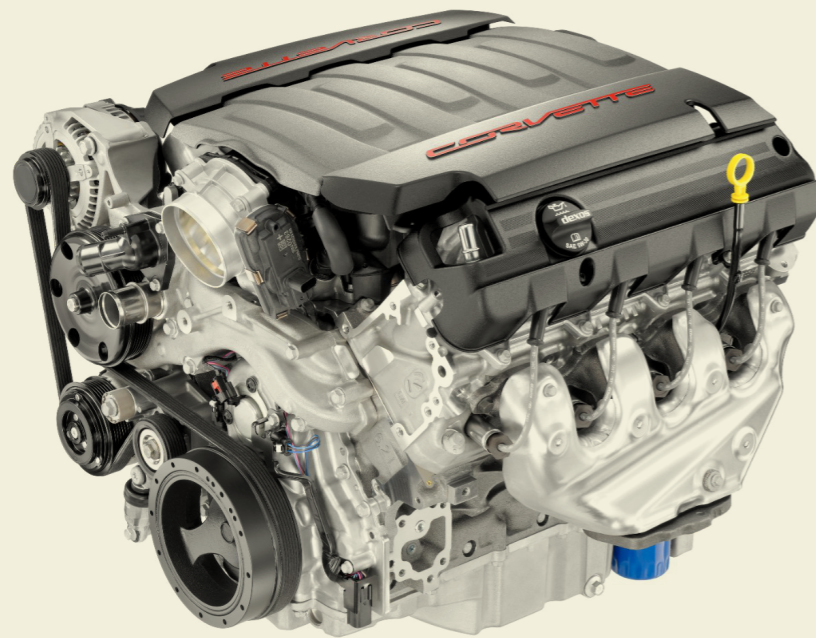
Srđan Krstić

with

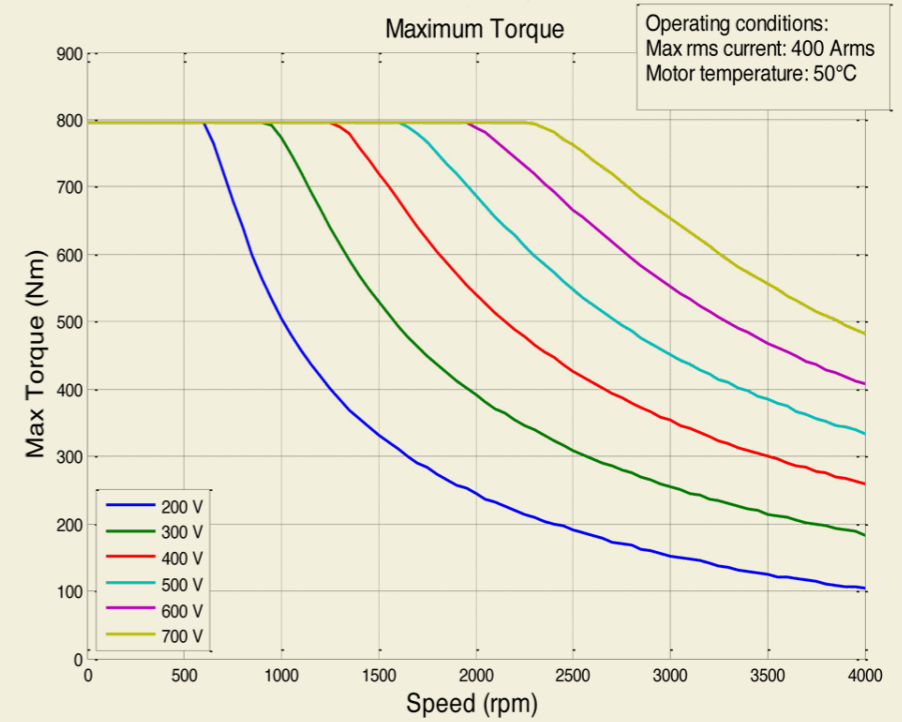
Domenico Bianculli, Carlo Ghezzi and Pierluigi San Pietro



System



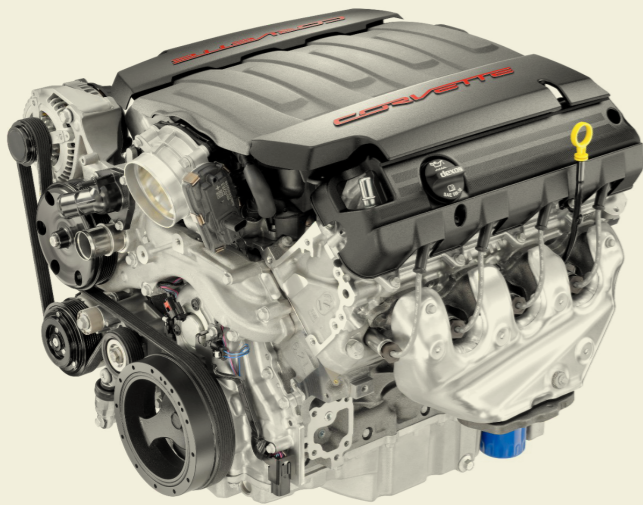
Specification



Execution Traces

Execution Traces

System

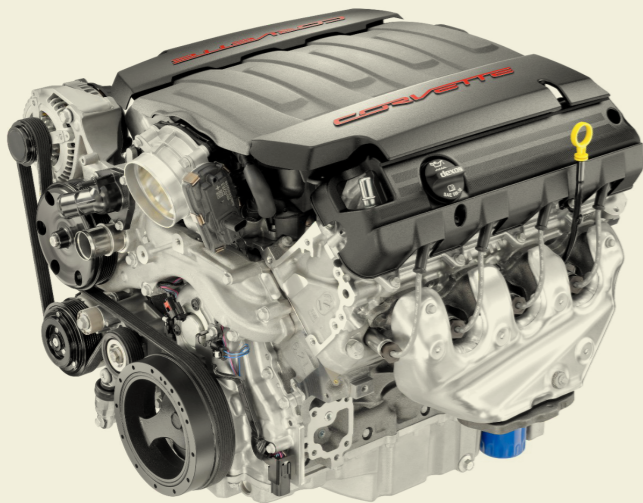


Execution Traces



Monitor

System

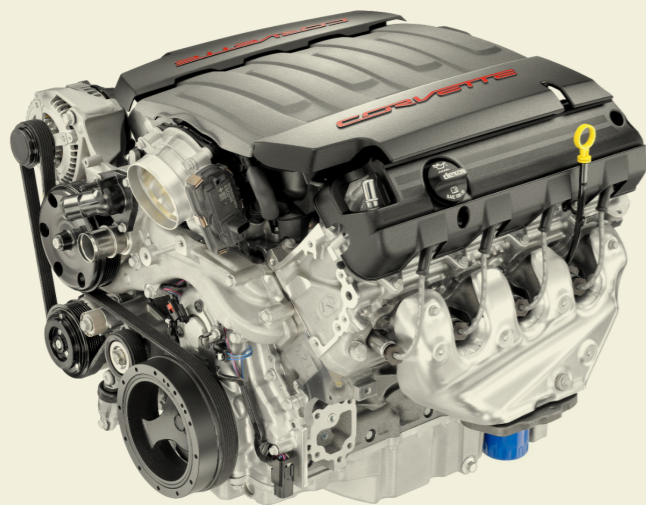


Execution Traces

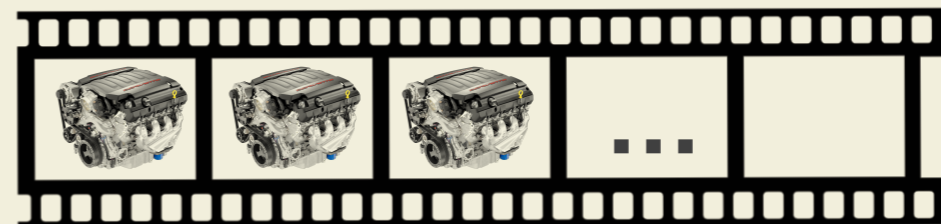


Monitor

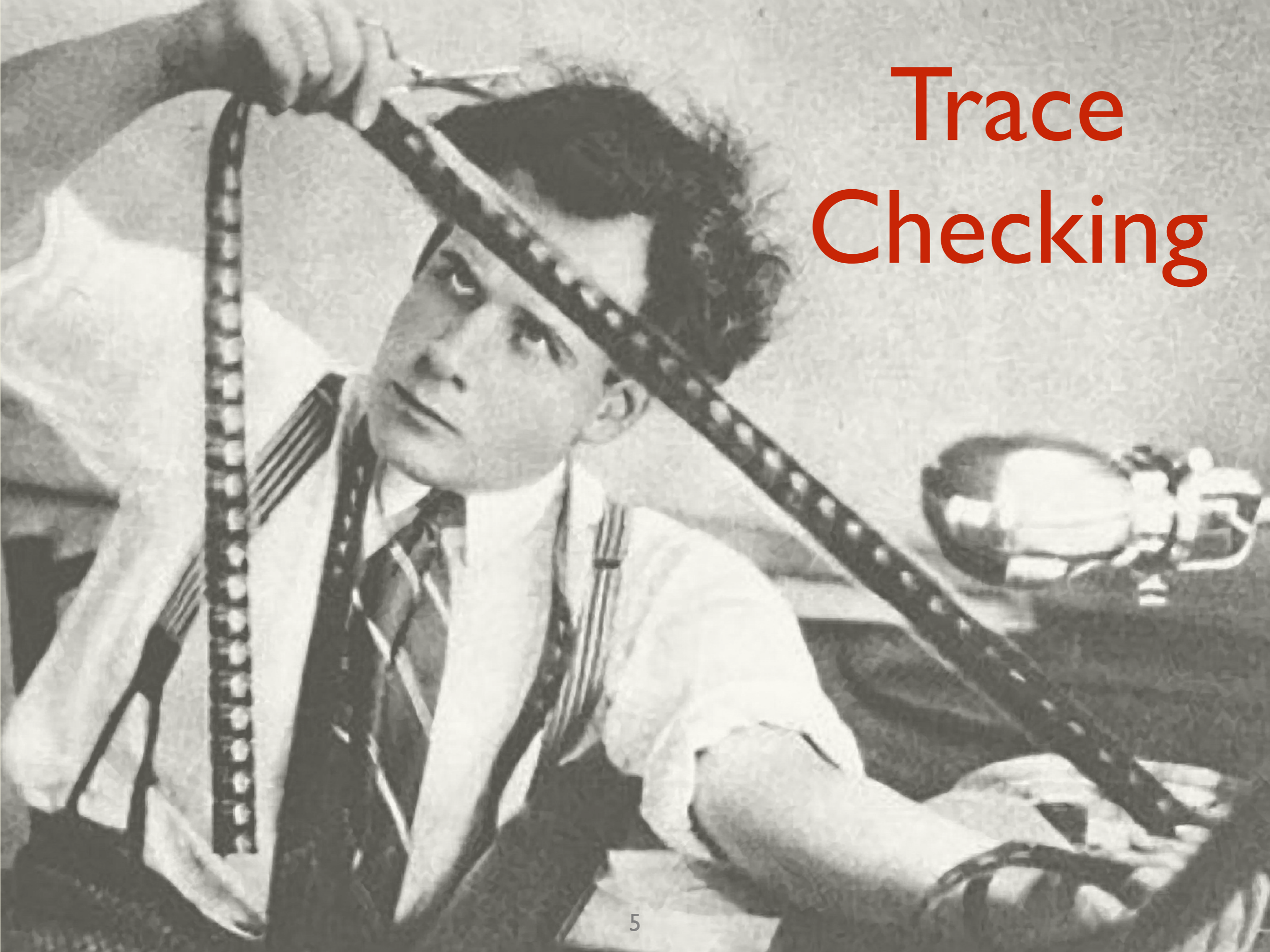
System



Execution trace

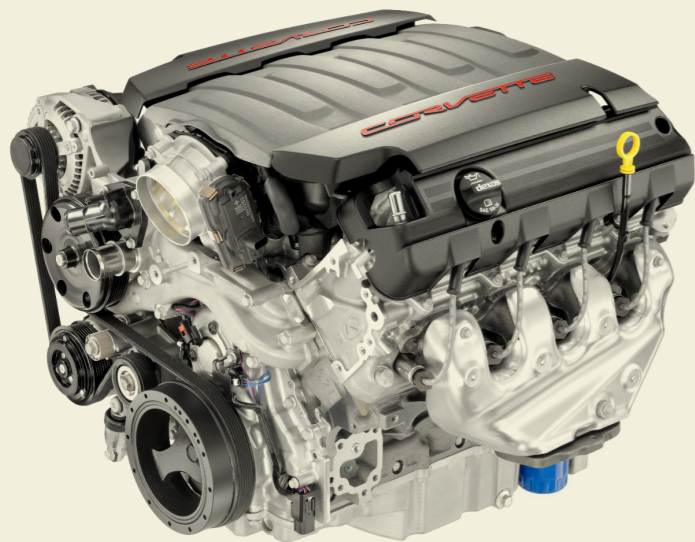


Trace Checking



Bounded Satisfiability Checking

System




Specification

φ

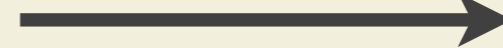
Bounded Satisfiability Checking

System

Specification

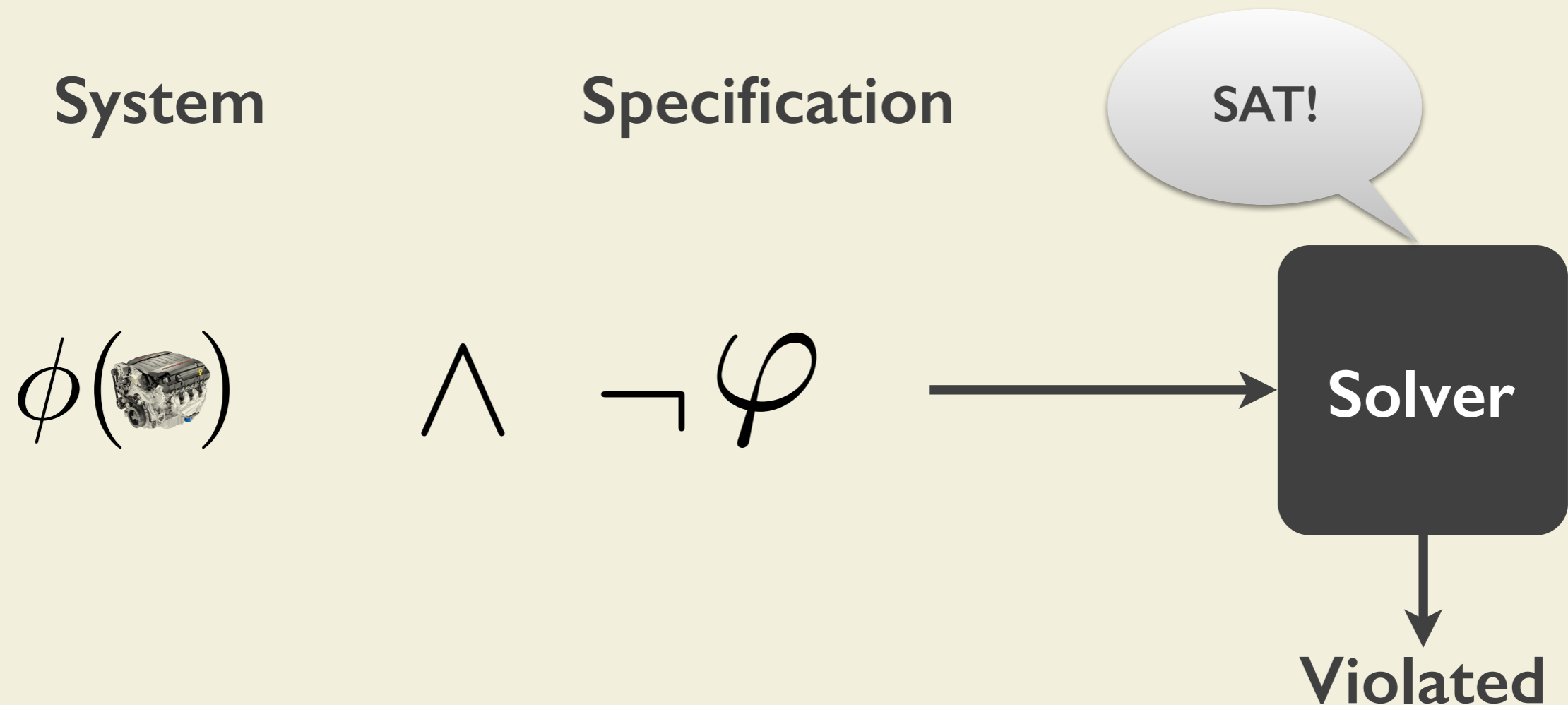
ϕ ()

$\wedge \neg \varphi$



Solver

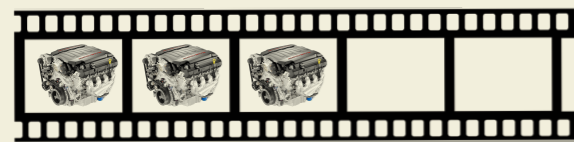
Bounded Satisfiability Checking



Bounded Satisfiability Checking

Execution trace

Temporal logic



$\wedge \neg \varphi$



Specification

SOLOIST

SOLOIST

SpecificatiOn Language fOr
service compoSitions inTeractions

SOLOIST

 $\neg p$

propositional

 $\phi U_{[3,17]} \psi$

metric temporal logic

 $\sigma^k_{<2} \phi$

**with additional
modalities**

SOLOIST

 $\neg p$

propositional

 $\phi U_{[3,17]} \psi$

metric temporal logic

 $\sigma^k_{<2} \phi$

with additional modalities

The Tale of SOLOIST: a Specification Language for Service Compositions Interactions

Domenico Bianculli¹ and Carlo Ghezzi² and Pierluigi San Pietro²

¹ University of Luxembourg - SnT Centre, Luxembourg
domenico.bianculli@uni.lu

² Politecnico di Milano - DEI - DEEP-SE Group, Italy
{carlo.ghezzi,pierluigi.sanpietro}@polimi.it

Abstract. Service-based applications are a new class of software systems that provide the basis for enterprises to build their information systems by following the principles of service-oriented architectures. These software systems are often realized by orchestrating remote, third-party services, to provide added-values applications that are called service compositions. The distributed ownership and the evolving nature of the services involved in a service composition make verification activities crucial. On a par with verification is also the problem of formally specifying the interactions—with third-party services—of service compositions, with the related issue of balancing expressiveness and support for automated verification.

This paper showcases SOLOIST, a specification language for formalizing the interactions of service compositions. SOLOIST has been designed with the primary objective of expressing the most significant specification patterns found in the specifications of service-based applications. The language is based on a many-sorted first-order metric temporal logic, extended with new temporal modalities that support aggregate operators for events occurring in a certain time window. We also show how, under certain assumptions, the language can be reduced to linear temporal logic, paving the way for using SOLOIST with established verification techniques, both at design time and at run time.

1 Introduction

Modern-age software engineering has to deal with novel kinds of software systems, which exhibit new features that often demand for rethinking and extending the traditional methodologies and the accompanying methods and techniques. One class of new software systems is constituted by *open-world* software [5], characterized by a dynamic and decentralized nature; service-based applications (SBAs) represent an example of this class of systems. SBAs are often defined as service compositions, obtained by orchestrating—with languages such as BPEL [2]—existing services, possibly offered by third-parties. This kind of applications has seen a wide adoption in enterprises, which nowadays develop their information systems using the principles of service orientation [20].

FACS 2012

Service Provisioning Patterns

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns

625 Industrial requirements
specification

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns

625 Industrial requirements
specification

290 Academic requirements
specification

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns

Counting the number of events

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns

Counting the number of events

Average number of events

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns

Counting the number of events

Average number of events

Maximum number of events

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns

Counting the number of events

Average number of events

Maximum number of events

Average response time

Specification Patterns from Research to Industry: A Case Study in Service-Based Applications

Domenico Bianculli
Faculty of Informatics
University of Lugano
Lugano, Switzerland
domenico.bianculli@usi.ch

Carlo Ghezzi
DEEPSE group - DEI
Politecnico di Milano
Milano, Italy
ghezzi@elet.polimi.it

Cesare Pautasso
Faculty of Informatics
University of Lugano
Lugano, Switzerland
cesare.pautasso@usi.ch

Patrick Senti
Information Technology
Credit Suisse AG
Zürich, Switzerland
patrick.senti@credit-suisse.com

Abstract—Specification patterns have proven to help developers to state precise system requirements, as well as formalize them by means of dedicated specification languages. Most of the past work has focused its applicability area to the specification of concurrent and real-time systems, and has been limited to a research setting. In this paper we present the results of our study on specification patterns for service-based applications (SBAs). The study focuses on industrial SBAs in the banking domain. We started by performing an extensive analysis of the usage of specification patterns in published research case studies — representing almost ten years of research in the area of specification, verification, and validation of SBAs. We then compared these patterns with a large body of specifications written by our industrial partner over a similar time period. The paper discusses the outcome of this comparison, indicating that some needs of the industry, especially in the area of requirements specification languages, are not fully met by current software engineering research.

Keywords—specification patterns; specification languages; requirements specifications; services

I. INTRODUCTION

The concept of *pattern* has been initially proposed in the domain of architecture by C. Alexander [1], to represent “the description of a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice”.

This idea of pattern has then been adopted in software engineering with the concept of *design patterns* [2], as reusable solutions for recurring problems in software design. Subsequently, the concept of design patterns has been embraced in different sub-domains of software engineering, from architectural patterns to reengineering patterns, including property specification patterns.

Property specification patterns [3] have been proposed in the late '90s in the context of finite-state verification, as a means to express recurring properties in a generalized form, which could be formalized in different specification languages, such as temporal logic. Specification patterns aimed at bridging the gap between finite-state verification

tools (e.g., model checkers) and practitioners, by providing the latter with a powerful instrument for writing down properties to be fed to a formal verification tool.

Given the origin of property specification patterns, most of past work has focused its applicability area to the specification (and the verification) of concurrent and real-time systems (see, for example, [4]), with limited applications outside the research setting.

In the last years, open software [5] systems such as service-based applications (SBAs) have emerged, introducing new engineering challenges due to their dynamic and decentralized nature. One of these research challenges is related to the specification, verification and validation of SBAs [6]. At the same time, service-oriented architectures (SOAs) have gained a lot of attention in enterprises, which started to adopt them for the integration of their information systems [7]. However, to the best of our knowledge, the research literature on specification, verification and validation of SBAs has presented limited evidence of its applicability to and suitability for industrial-level case studies.

One of the questions that we asked ourselves during our research is whether existing requirements specification languages are expressive enough to formalize common requirements specifications used in industry. In particular, we are interested in evaluating the use of specification patterns for expressing properties of industrial SBAs, to assess if existing and well-known specification patterns are adequate or not. If this is not the case, our goal is to gather substantial evidence for new specification patterns and/or language constructs required to support their practical use in industrial settings.

In this paper we present the results of our study on the use of specification patterns in SBAs. The study has been performed by analyzing the requirements specifications of two sets of case studies. One set was composed of case studies extracted from research papers in the area of specification, verification and validation of SBAs, which appeared in the main publishing venues of software engineering and service-oriented computing within the last 10 years. The other set was composed of case studies corresponding to

ICSE 2012

Service Provisioning Patterns in SOLOIST

Counting the number of events

Average number of events

Maximum number of events

Average response time

Service Provisioning Patterns in SOLOIST

Counting the number of events

$$\mathcal{C}_{\bowtie n}^K(\phi)$$

Average number of events

$$\mathcal{U}_{\bowtie n}^{K,h}(\phi)$$

Maximum number of events

$$\mathcal{M}_{\bowtie n}^{K,h}(\phi)$$

Average response time

$$\mathcal{D}_{\bowtie n}^K(\phi, \phi)$$

SOLOIST

Propositional

$$p \mid \neg\phi \mid \phi \wedge \phi$$

Metric Temporal Logic

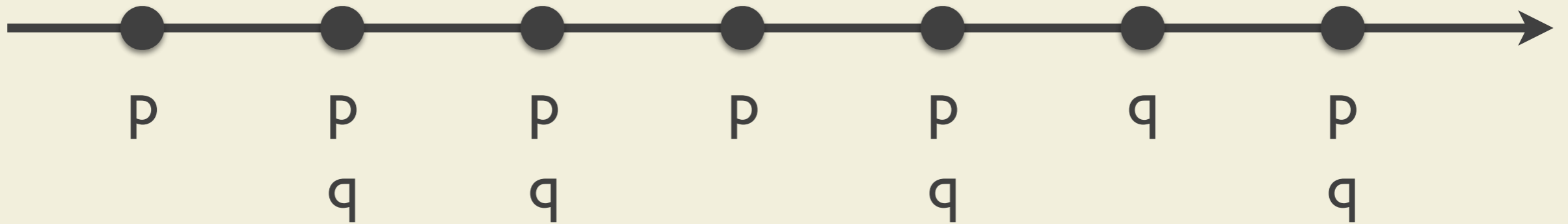
$$\phi U_I \phi \mid \phi S_I \phi$$

with aggregating modalities

$$\mathcal{E}_{\boxtimes n}^K(\phi) \mid \mathcal{U}_{\boxtimes n}^{K,h}(\phi) \mid \mathcal{M}_{\boxtimes n}^{K,h}(\phi) \mid \mathcal{D}_{\boxtimes n}^K(\phi, \phi)$$

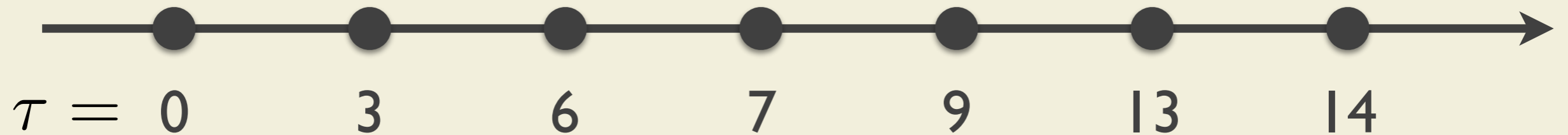
SOLOIST

Model: a timed ω -word



SOLOIST

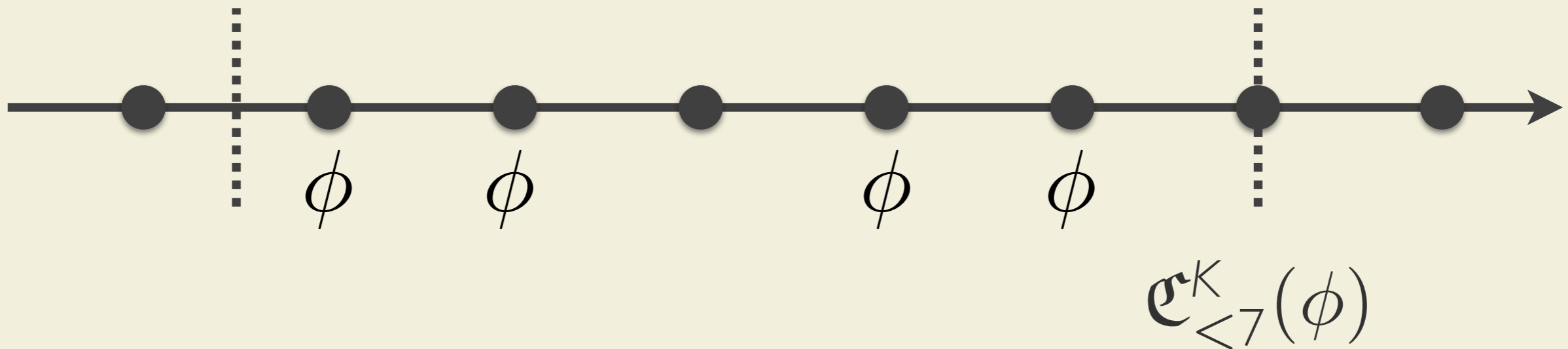
Model: a timed ω -word



Counting Modality

$\tau - K$

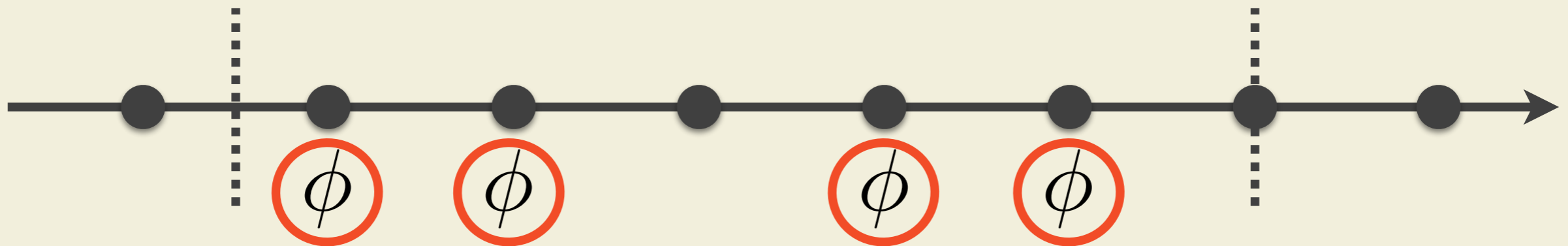
τ



Counting Modality

$\tau - K$

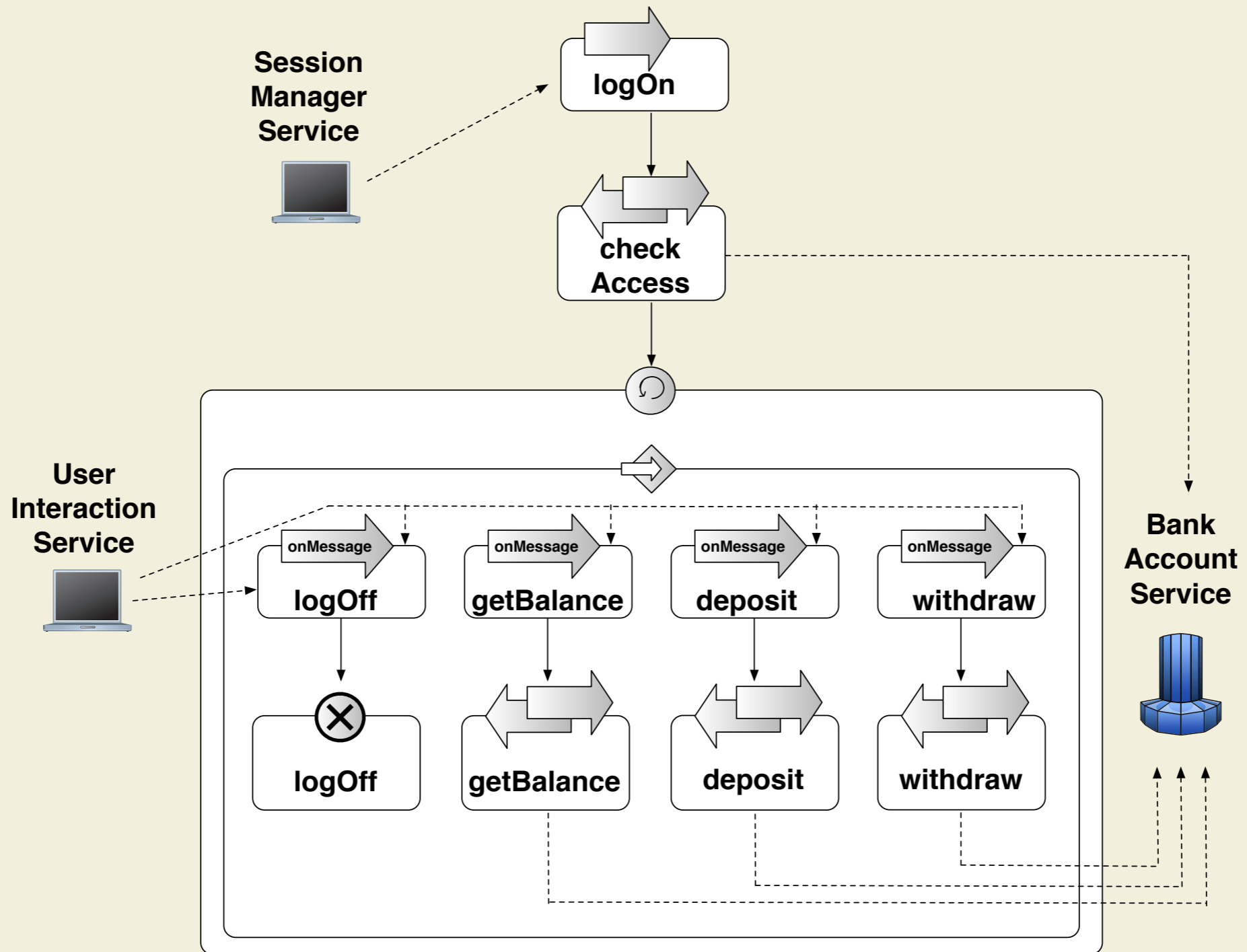
τ



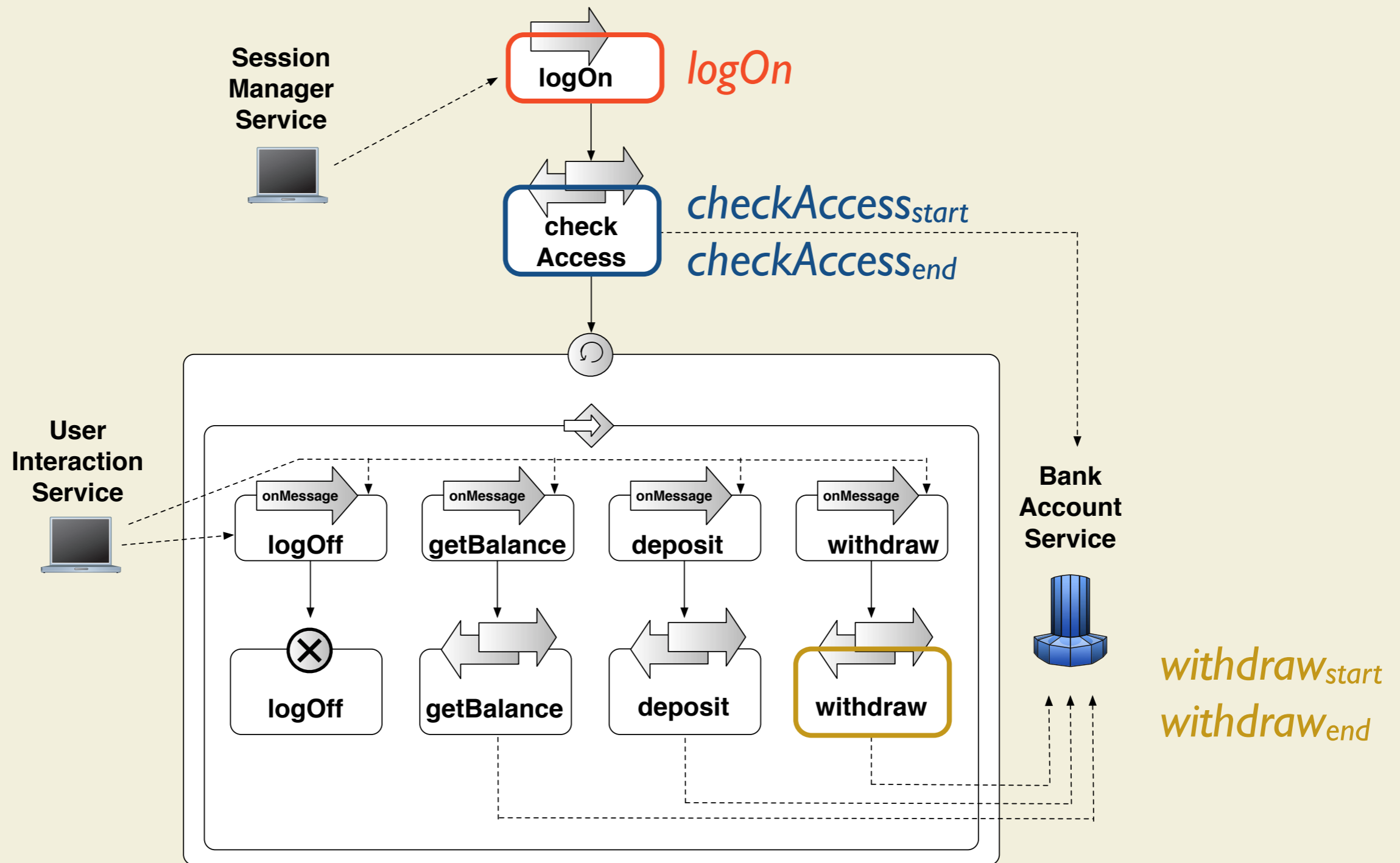
$c_{<7}^K(\phi)$

$4 < 7$

Example



Example



SOLOIST in Action

Counting Modality

“The number of **withdrawal** operations performed within **10 minutes** before customer **logs off** is **less than or equal to 3**”

$$G(\text{logOff} \rightarrow \mathop{e}_{\leq 3}^{600}(\text{withdraw}_{\text{end}}))$$

SOLOIST in Action

Maximum Modality

“The maximum number of **balance inquiries** is restricted to **at most 2 per minute** within **10 minutes** before customer **logs off**”

$$G(\text{logOff} \rightarrow \mathfrak{M}_{\leq 2}^{600, 60}(\text{getBalance}_{\text{end}}))$$

SOLOIST in Action

Average Distance Modality

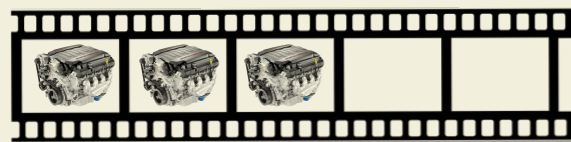
“The average response time of operation **checkAccess** provided by the BankAccount service is always **less than 5** seconds within any **15 minute** time window”

$$G(\mathcal{D}_{<5}^{900} \{ (checkAccess_{start}, checkAccess_{end}) \})$$

Bounded Satisfiability Checking

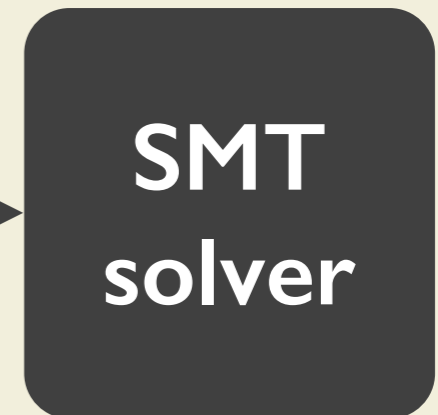
Execution trace

SOLOIST



\wedge

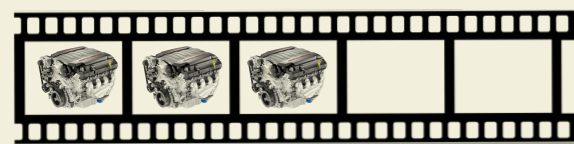
$\neg \varphi$



Bounded Satisfiability Checking

Execution trace

SOLOIST



\wedge

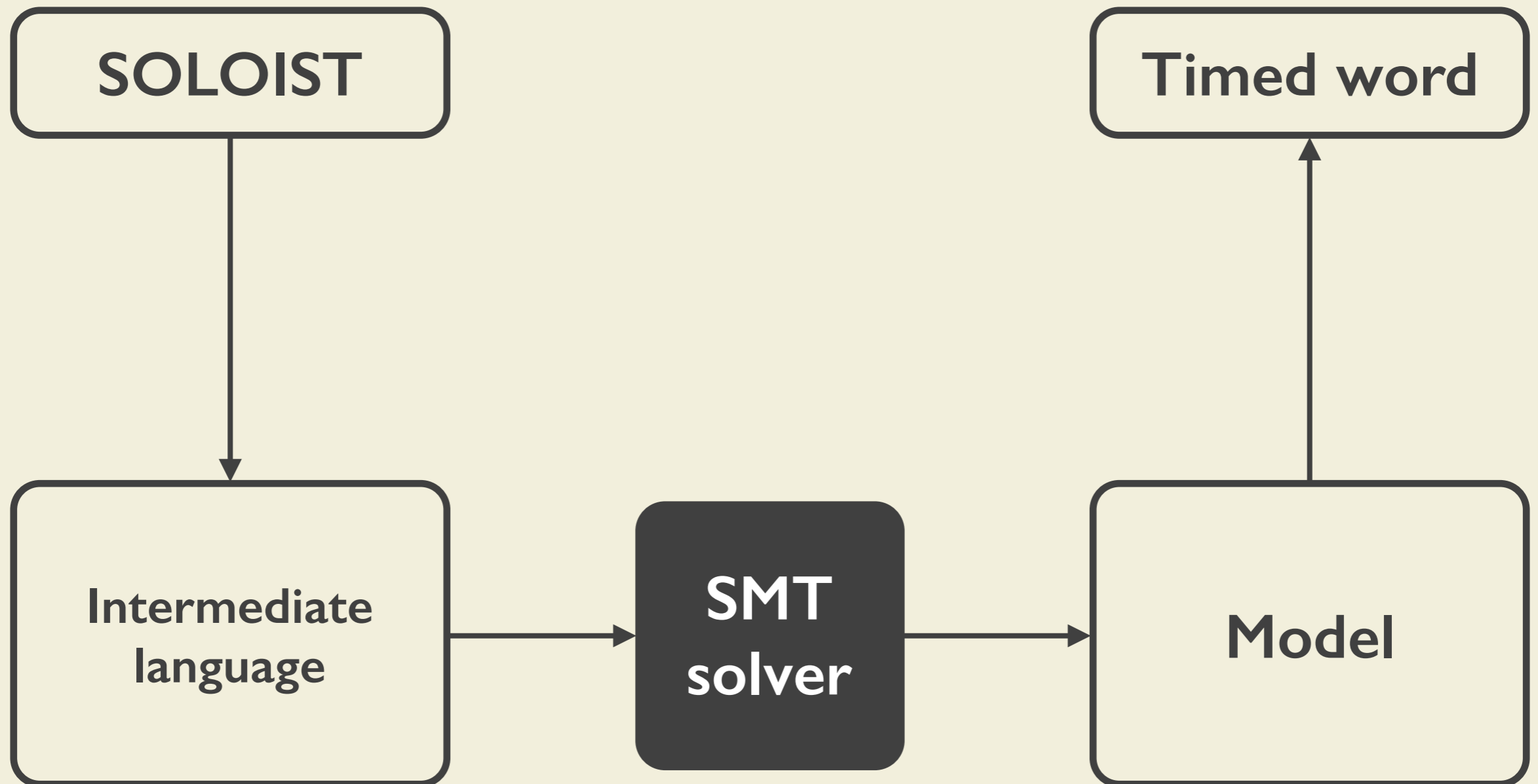
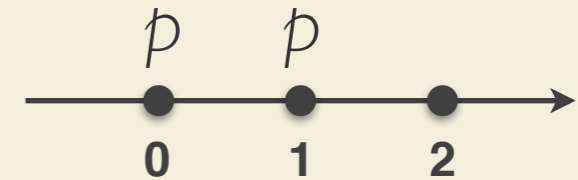
$\neg \varphi$



I don't speak SOLOIST!

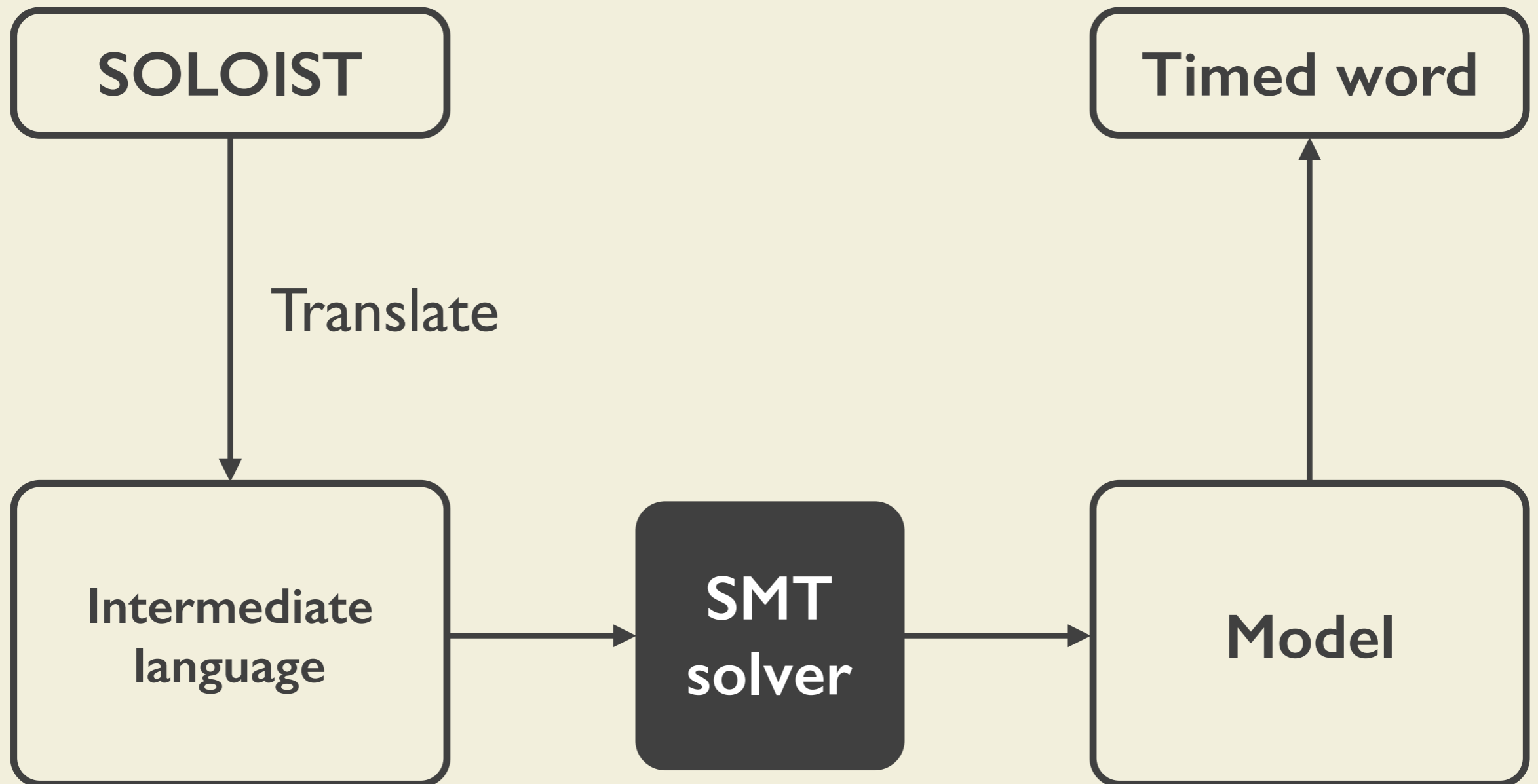
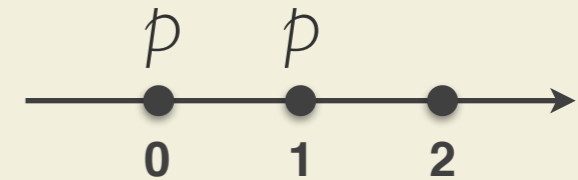
SOLOIST Satisfiability

$$e_{<5}^{100}(p)$$



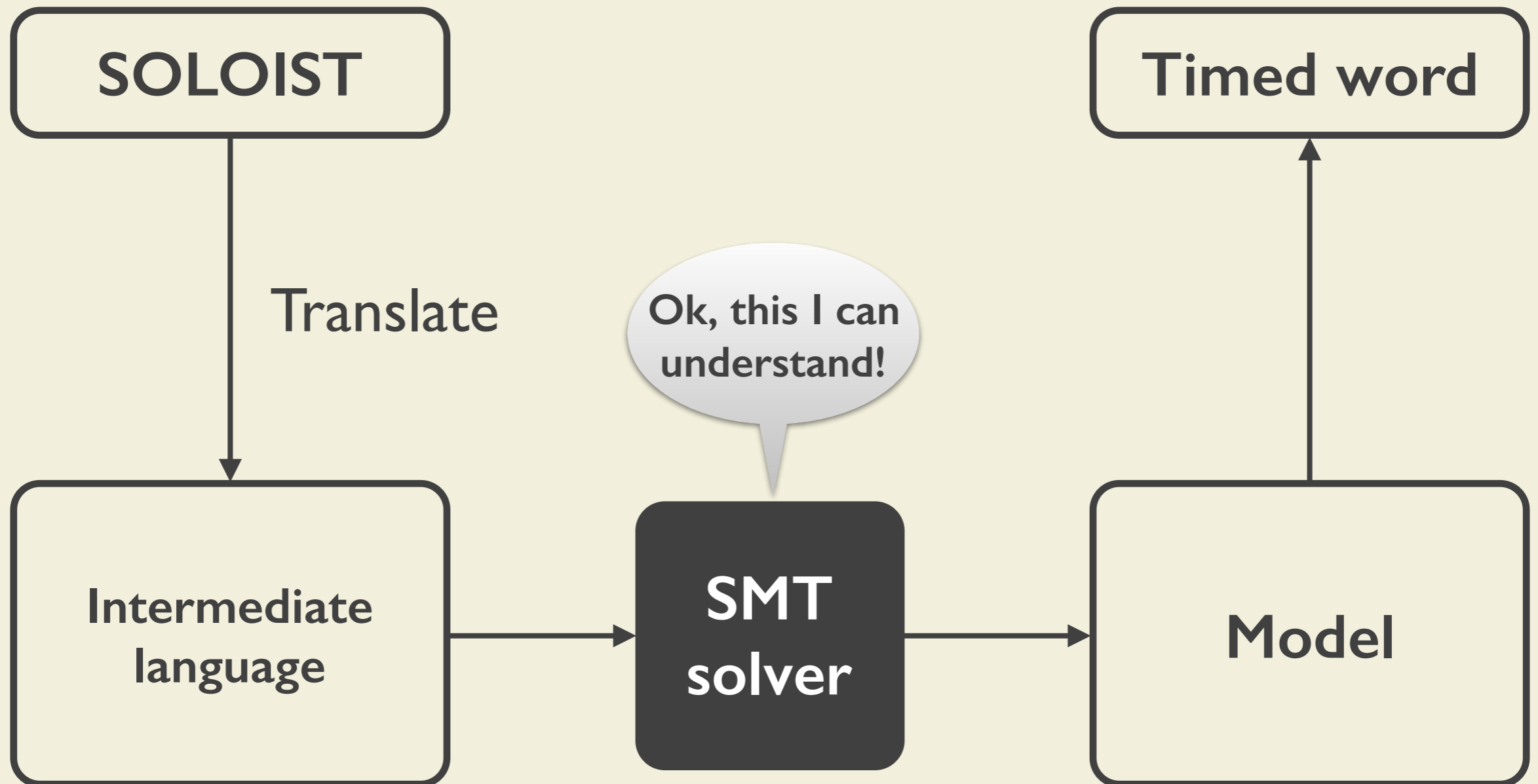
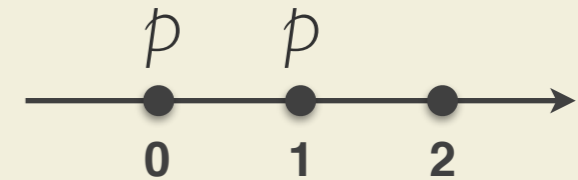
SOLOIST Satisfiability

$$e_{<5}^{100}(p)$$



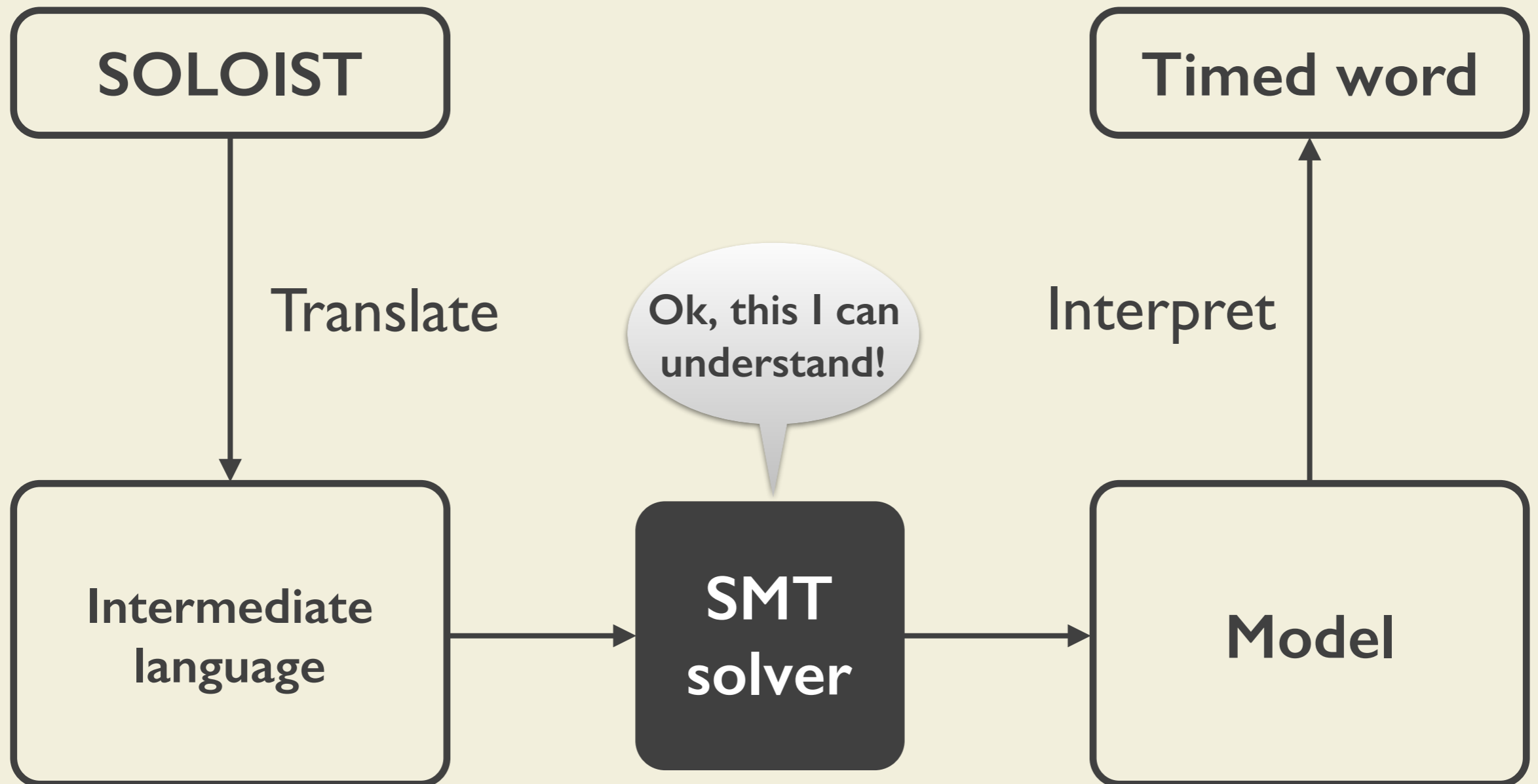
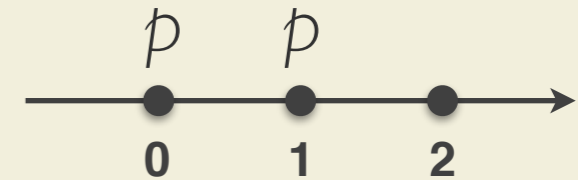
SOLOIST Satisfiability

$$e_{<5}^{100}(p)$$



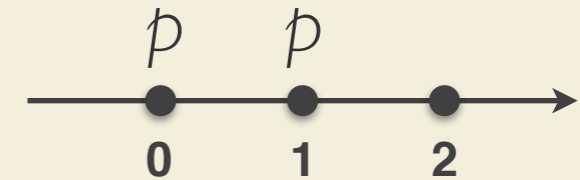
SOLOIST Satisfiability

$$e_{<5}^{100}(p)$$



SOLOIST Satisfiability

$$e_{<5}^{100}(p)$$



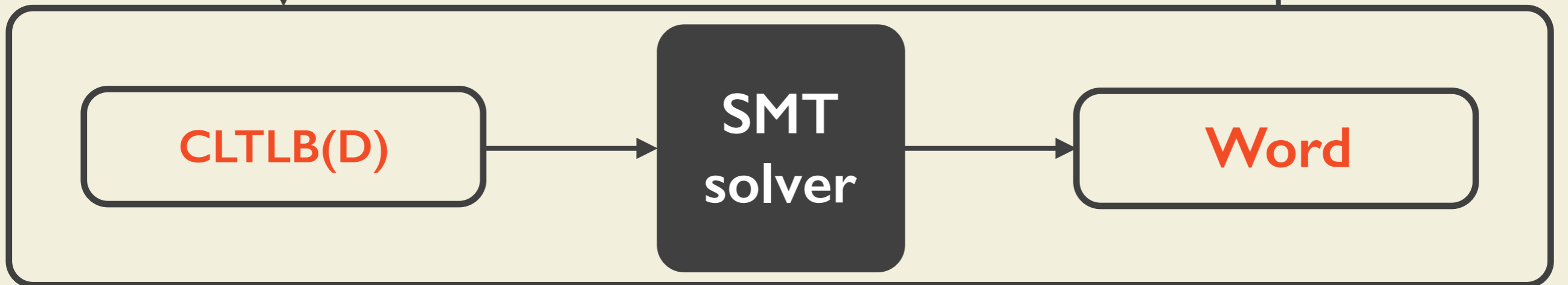
SOLOIST

Timed word

Translate

Interpret

ZOT



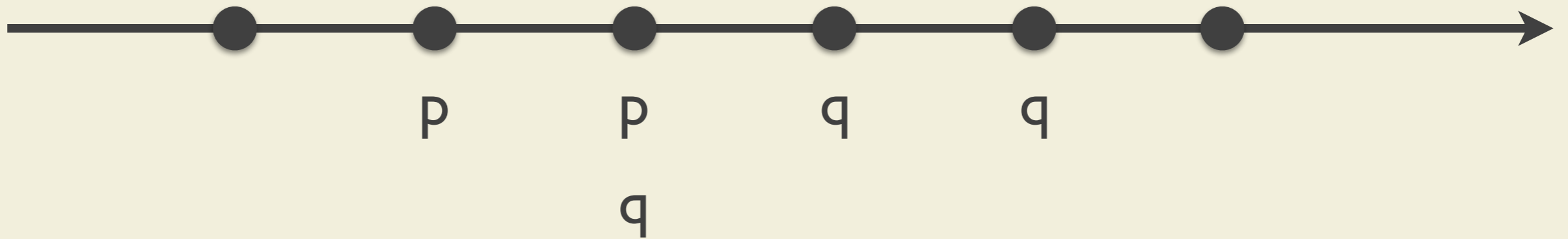
CLTLB(D)

CLTLB(D)

CLTLB(D)

Linear Temporal Logic

$$G(p \rightarrow XX(q))$$

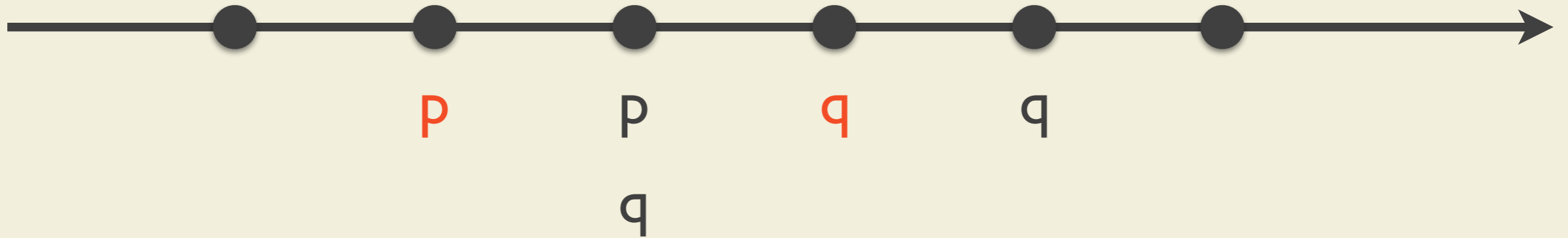


“It is always true that if **p** occurs then **q** occurs 2 positions afterwards”

CLTLB(D)

Linear Temporal Logic

$$G(p \rightarrow XX(q))$$

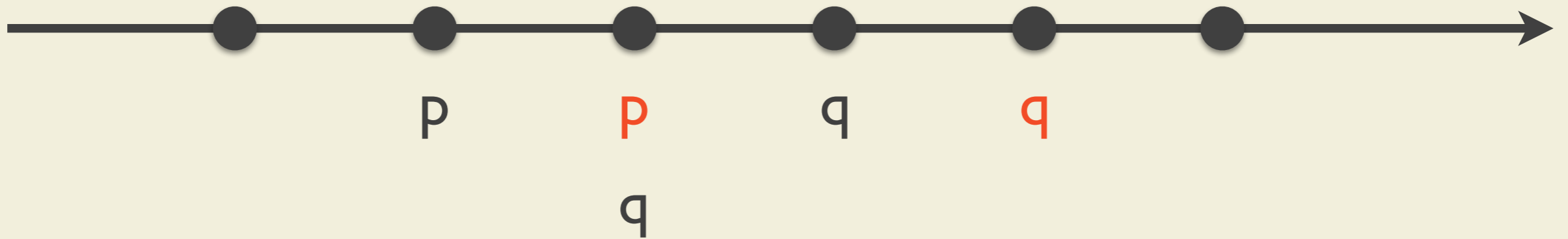


“It is always true that if **p** occurs then **q** occurs 2 positions afterwards”

CLTLB(D)

Linear Temporal Logic

$$G(p \rightarrow XX(q))$$

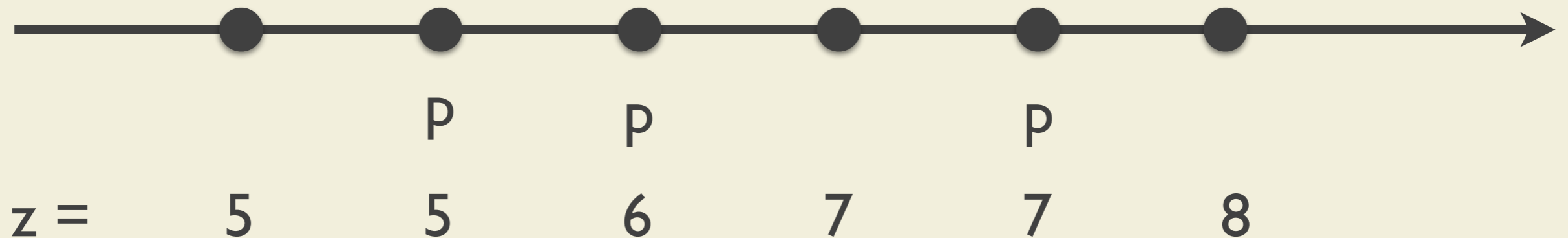


“It is always true that if **p** occurs then **q** occurs 2 positions afterwards”

CLTLB(D)

Constraint Linear Temporal Logic (over constraint system D)

$$G(p \leftrightarrow X(z) = z + 1)$$

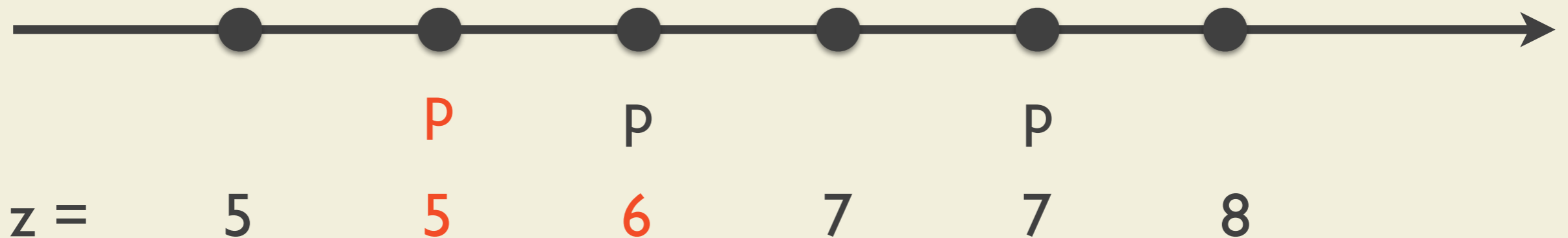


“There is **p** if and only if variable **z** is incremented by 1 in the next position”

CLTLB(D)

Constraint Linear Temporal Logic (over constraint system D)

$$G(p \leftrightarrow X(z) = z + 1)$$

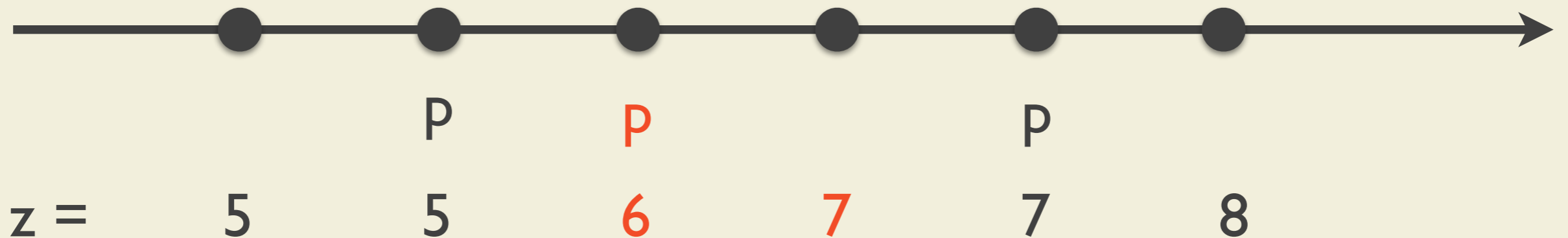


“There is p if and only if variable z is incremented by 1 in the next position”

CLTLB(D)

Constraint Linear Temporal Logic (over constraint system D)

$$G(p \leftrightarrow X(z) = z + 1)$$

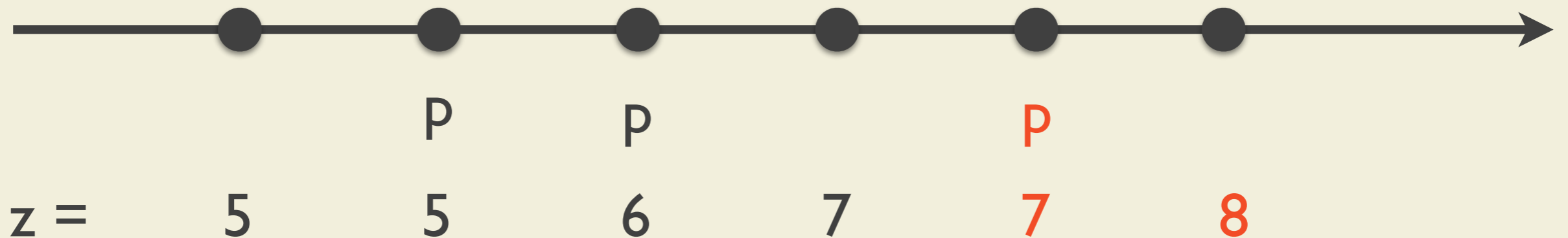


“There is p if and only if variable z is incremented by 1 in the next position”

CLTLB(D)

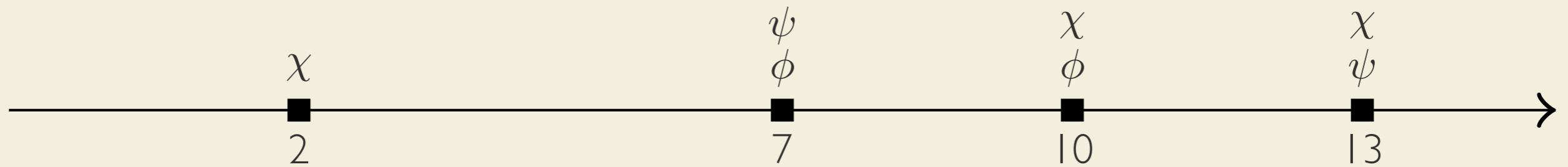
Constraint Linear Temporal Logic (over constraint system D)

$$G(p \leftrightarrow X(z) = z + 1)$$

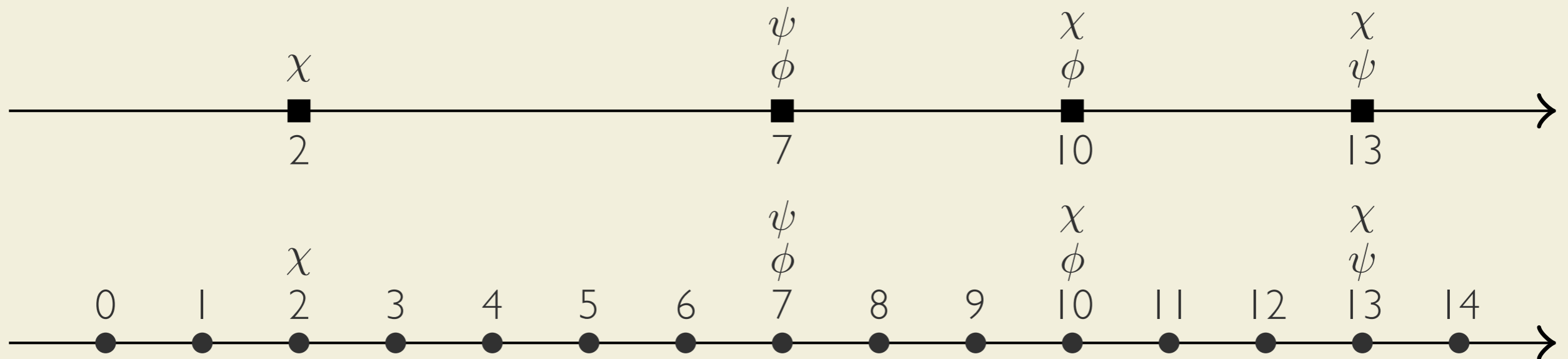


“There is p if and only if variable z is incremented by 1 in the next position”

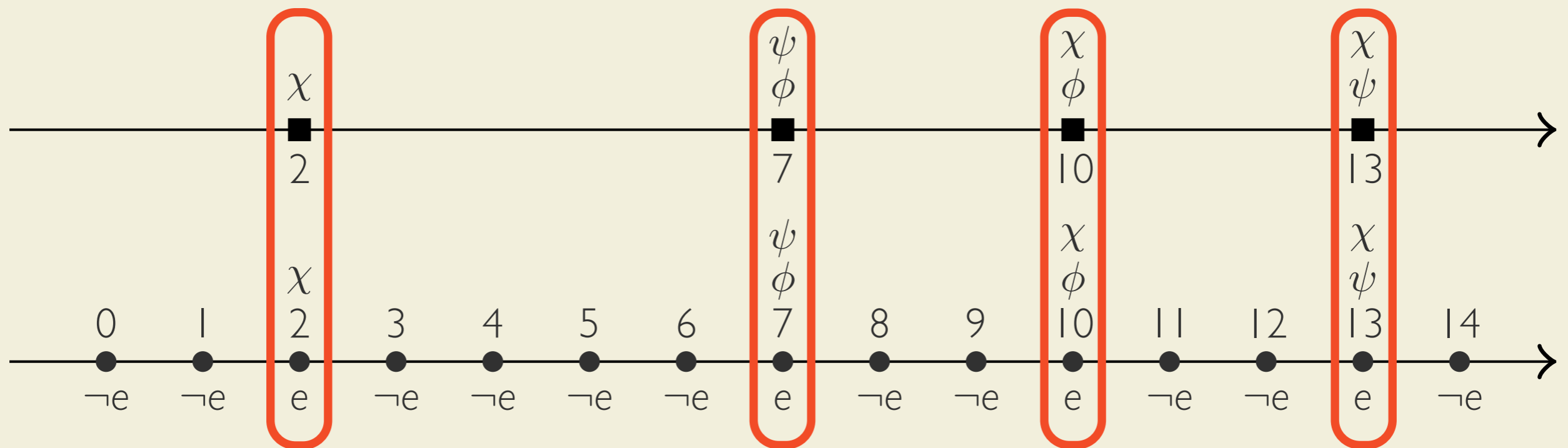
Interpretation of the Model



Interpretation of the Model



Interpretation of the Model



Encoding ρ of the Formula

Positive Normal Form

$$\{\wedge, \vee, U_I, R_I, S_I, T_I, \mathcal{E}_{\boxtimes n}^K, \mathcal{U}_{\boxtimes n}^{K,h}, \mathcal{M}_{\boxtimes n}^{K,h}, \mathcal{D}_{\boxtimes n}^K\} \cup \Pi \cup \bar{\Pi}$$

ρ (Boolean Formula)

$$\rho(p) \equiv p, p \in \Pi$$

$$\rho(\neg p) \equiv \neg p, p \in \Pi$$

$$\rho(\phi \wedge \psi) \equiv \rho(\phi) \wedge \rho(\psi)$$

$$\rho(\phi \vee \psi) \equiv \rho(\phi) \vee \rho(\psi)$$

ρ (Temporal Formula)

$$\rho(\phi U_I \psi) \equiv (\neg e \vee \rho(\phi)) U_I (e \wedge \rho(\psi))$$

ρ (Temporal Formula)

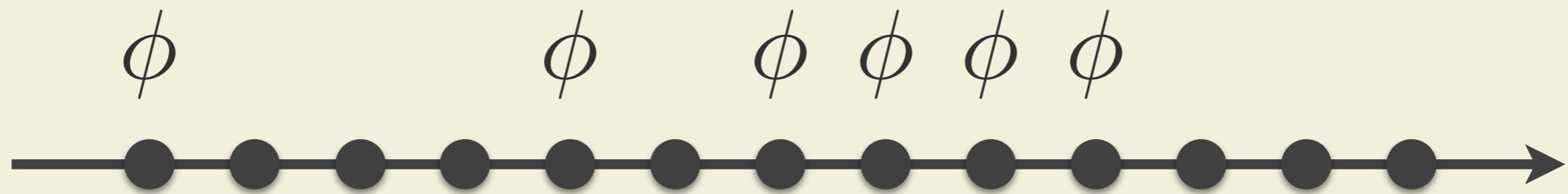
$$\rho(\phi \mathbf{U}_I \psi) \equiv (\neg e \vee \rho(\phi)) \mathbf{U}_I (e \wedge \rho(\psi))$$

$$\rho(\phi \mathbf{S}_I \psi) \equiv (\neg e \vee \rho(\phi)) \mathbf{S}_I (e \wedge \rho(\psi))$$

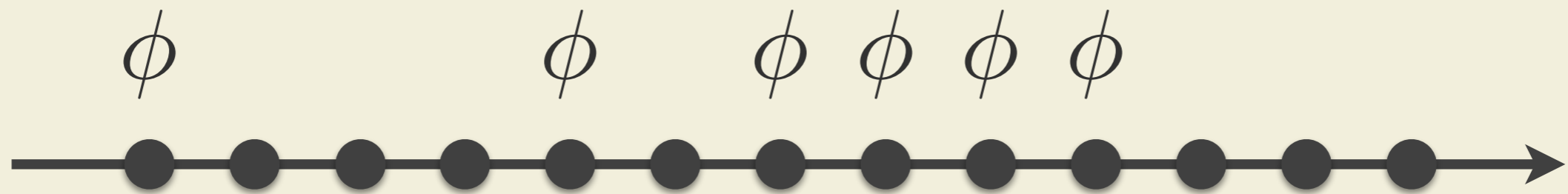
$$\rho(\phi \mathbf{R}_I \psi) \equiv (e \wedge \rho(\phi)) \mathbf{R}_I (\neg e \vee \rho(\psi))$$

$$\rho(\phi \mathbf{T}_I \psi) \equiv (e \wedge \rho(\phi)) \mathbf{T}_I (\neg e \vee \rho(\psi))$$

Counting Modality Counters



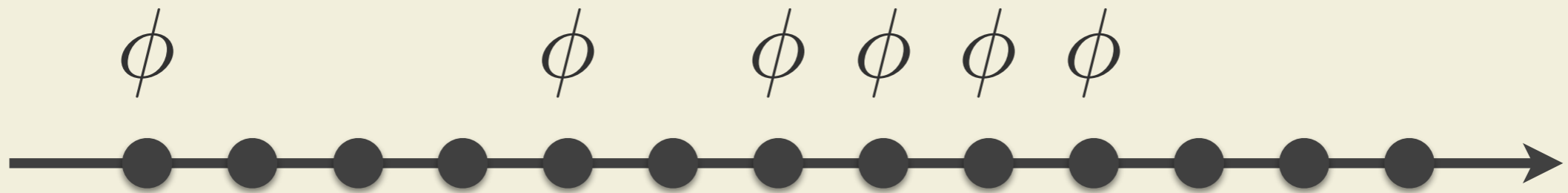
Counting Modality Counters



$C_\phi =$

Counting Modality Counters

$$c_{\phi} = 0$$

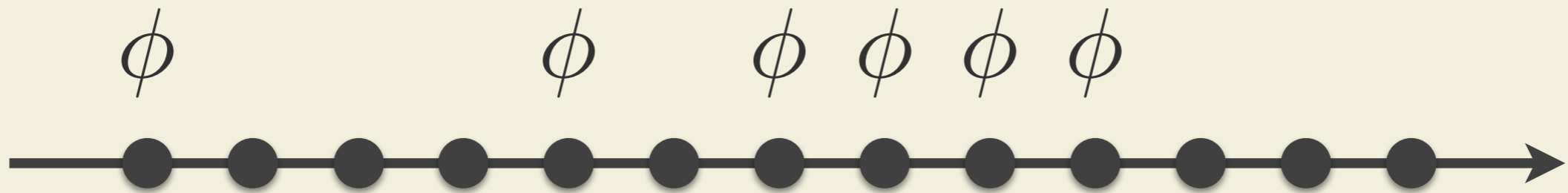


$$c_{\phi} = 0$$

Counting Modality Counters

$$c_\phi = 0$$

$$\mathbf{G}((e \wedge \phi) \rightarrow \mathbf{X}(c_\phi) = c_\phi + 1)$$



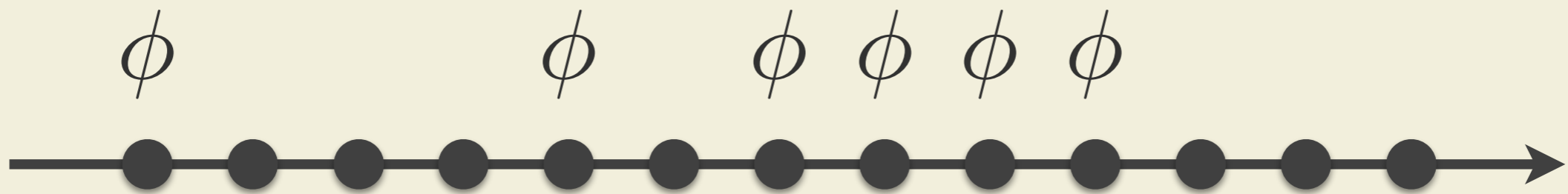
$$c_\phi = 0 \quad |$$

Counting Modality Counters

$$c_\phi = 0$$

$$\mathbf{G}((e \wedge \phi) \rightarrow \mathbf{X}(c_\phi) = c_\phi + 1)$$

$$\mathbf{G}((\neg e \vee \neg \phi) \rightarrow \mathbf{X}(c_\phi) = c_\phi)$$



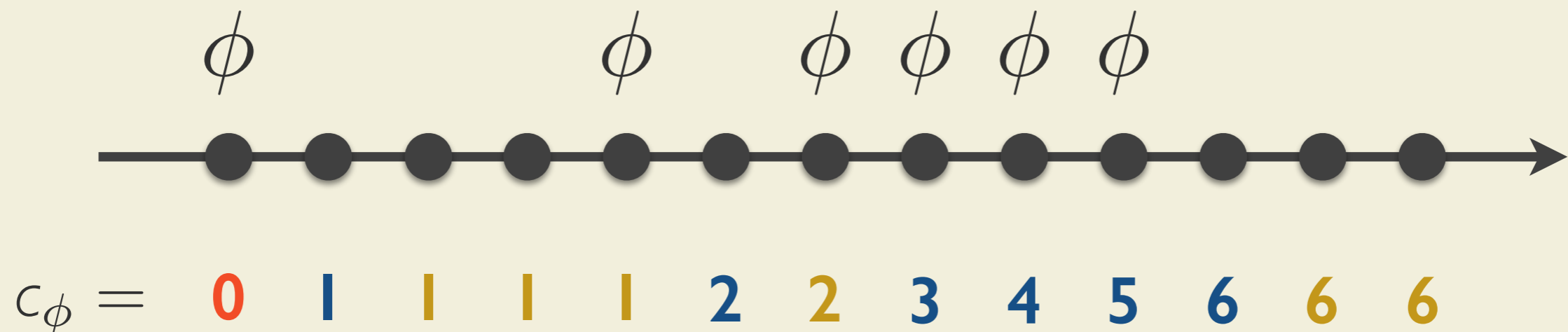
$$c_\phi = 0 \quad | \quad |$$

Counting Modality Counters

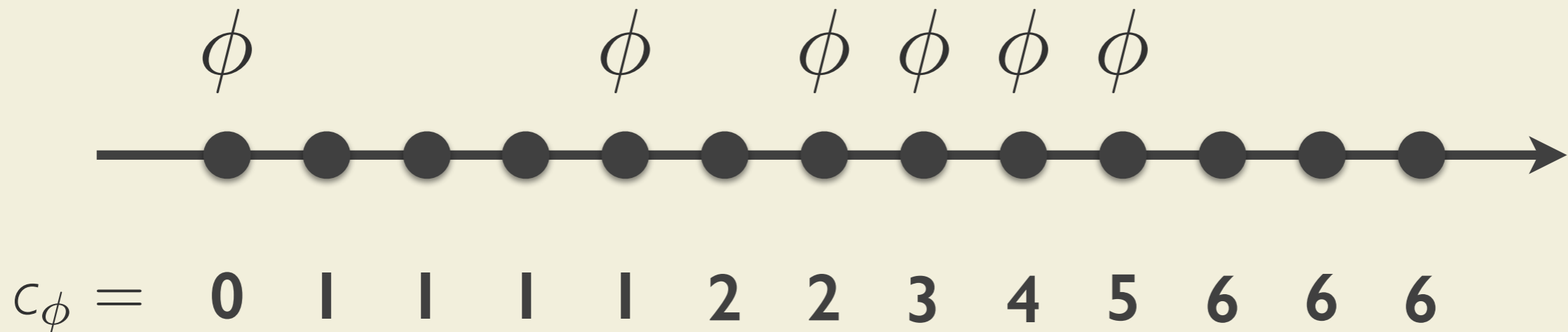
$$c_\phi = 0$$

$$\mathbf{G}((e \wedge \phi) \rightarrow \mathbf{X}(c_\phi) = c_\phi + 1)$$

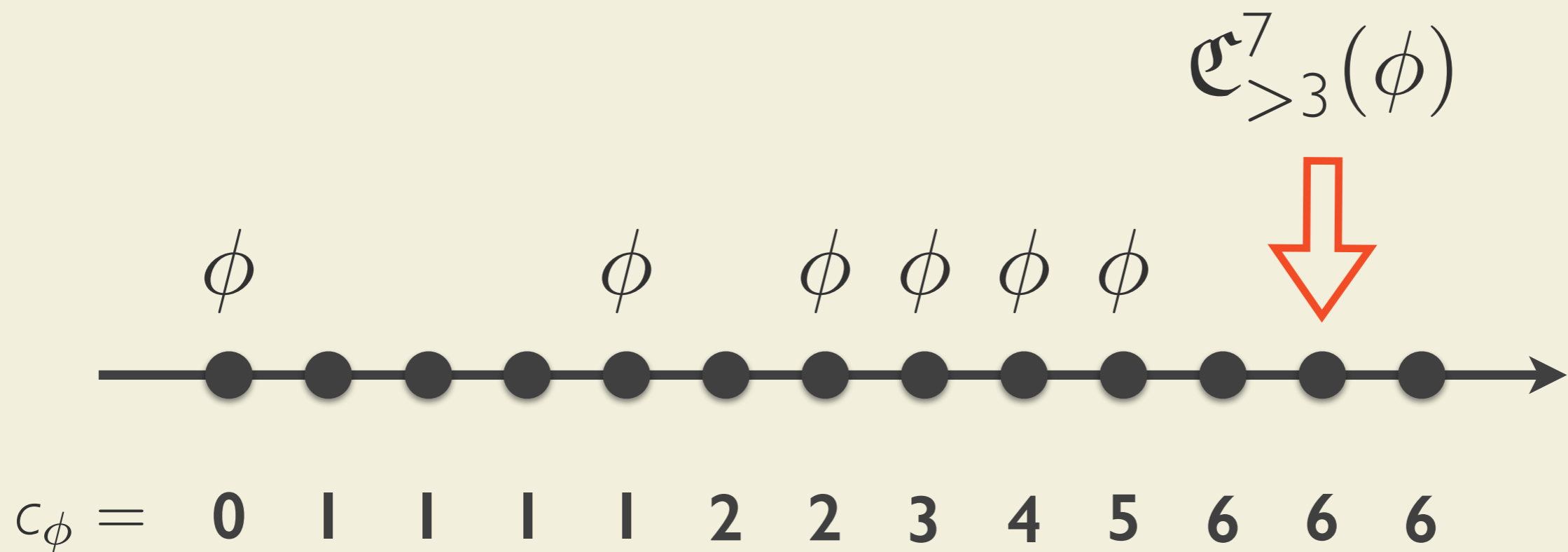
$$\mathbf{G}((\neg e \vee \neg \phi) \rightarrow \mathbf{X}(c_\phi) = c_\phi)$$



ρ (Counting Modality Formula)



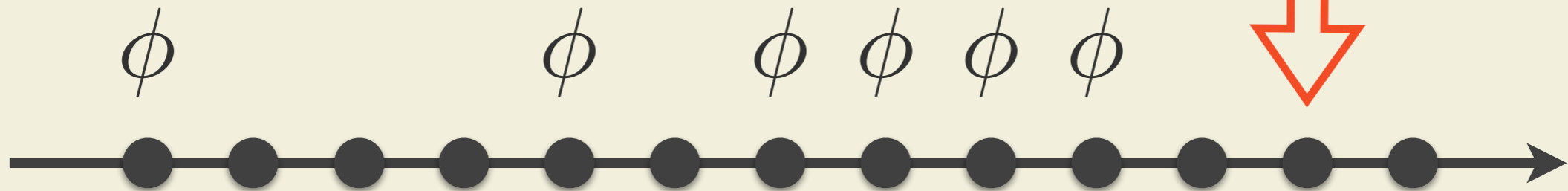
ρ (Counting Modality Formula)



ρ (Counting Modality Formula)

$$\rho(\mathfrak{C}_{>3}^7(\phi))$$

$$\mathfrak{C}_{>3}^7(\phi)$$

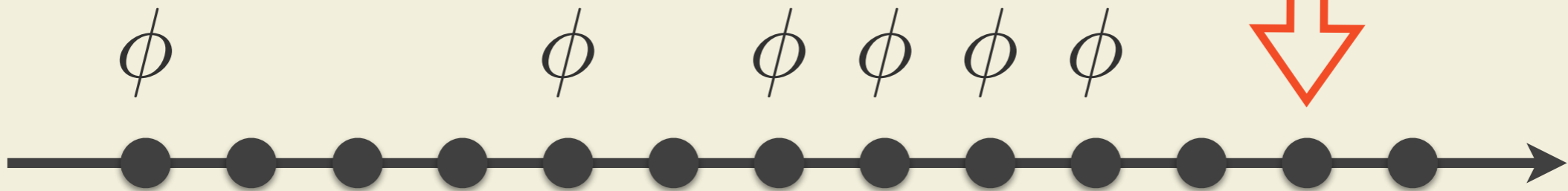


$$c_\phi = 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 6 \quad 6$$

ρ (Counting Modality Formula)

$$\rho(\mathfrak{C}_{>3}^7(\phi)) \equiv X(c_\phi) - Y^6(c_\phi) > 3$$

$\mathfrak{C}_{>3}^7(\phi)$

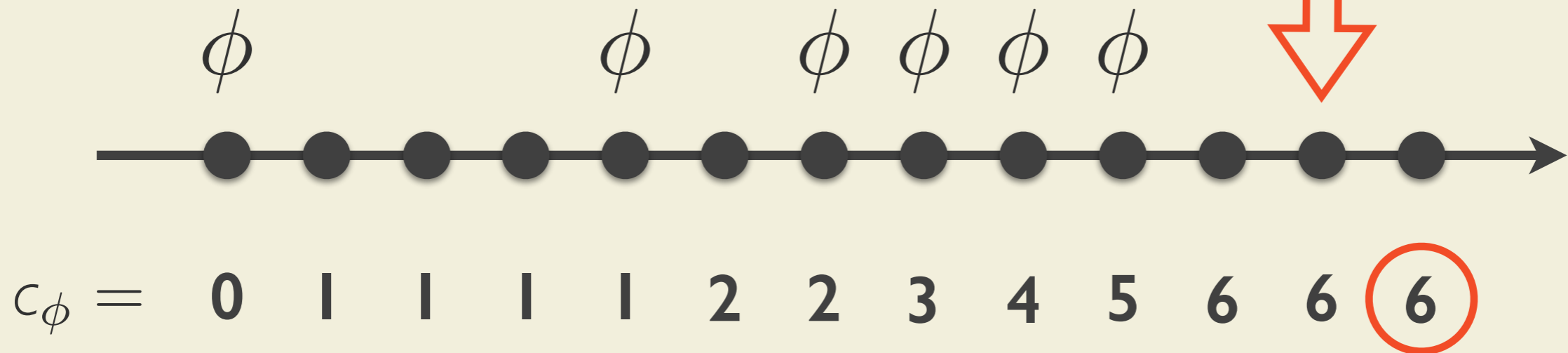


$c_\phi =$ 0 1 1 1 1 2 2 3 4 5 6 6 6

ρ (Counting Modality Formula)

$$\rho(\mathfrak{C}_{>3}^7(\phi)) \equiv \boxed{X(c_\phi)} - Y^6(c_\phi) > 3$$

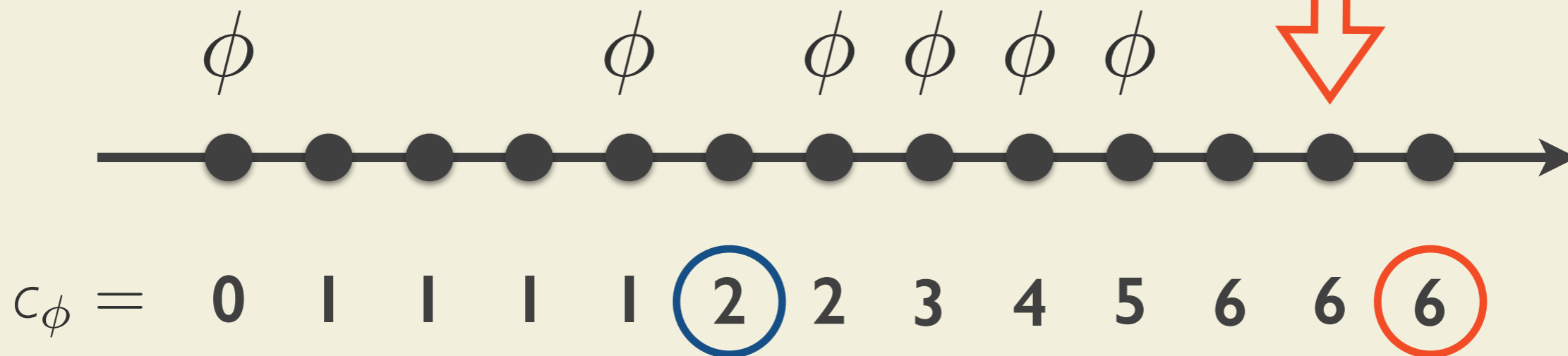
$\mathfrak{C}_{>3}^7(\phi)$



ρ (Counting Modality Formula)

$$\rho(\mathfrak{C}_{>3}^7(\phi)) \equiv X(c_\phi) - Y^6(c_\phi) > 3$$

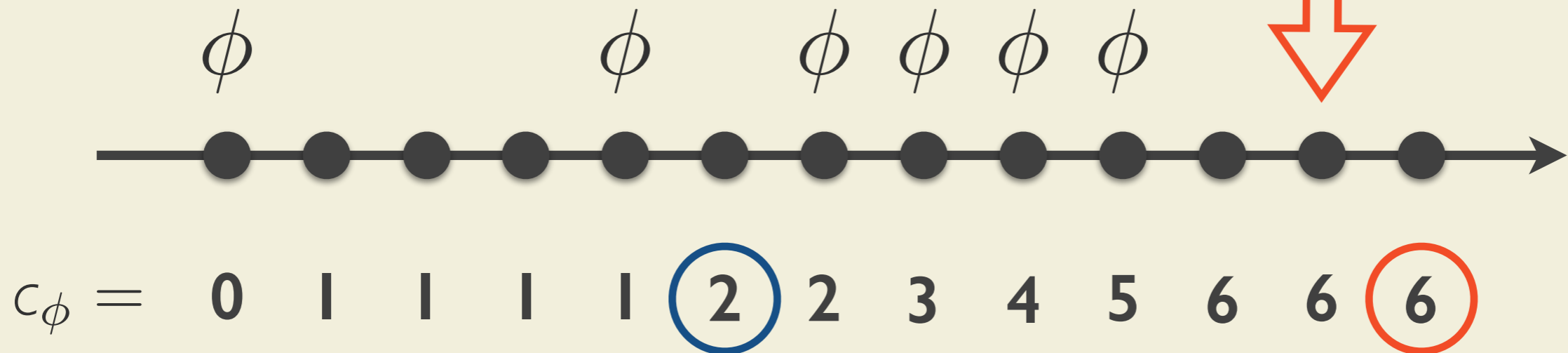
$\mathfrak{C}_{>3}^7(\phi)$



ρ (Counting Modality Formula)

$$\rho(\mathfrak{C}_{>3}^7(\phi)) \equiv X(c_\phi) - Y^6(c_\phi) > 3$$

$\mathfrak{C}_{>3}^7(\phi)$



$$4 > 3$$

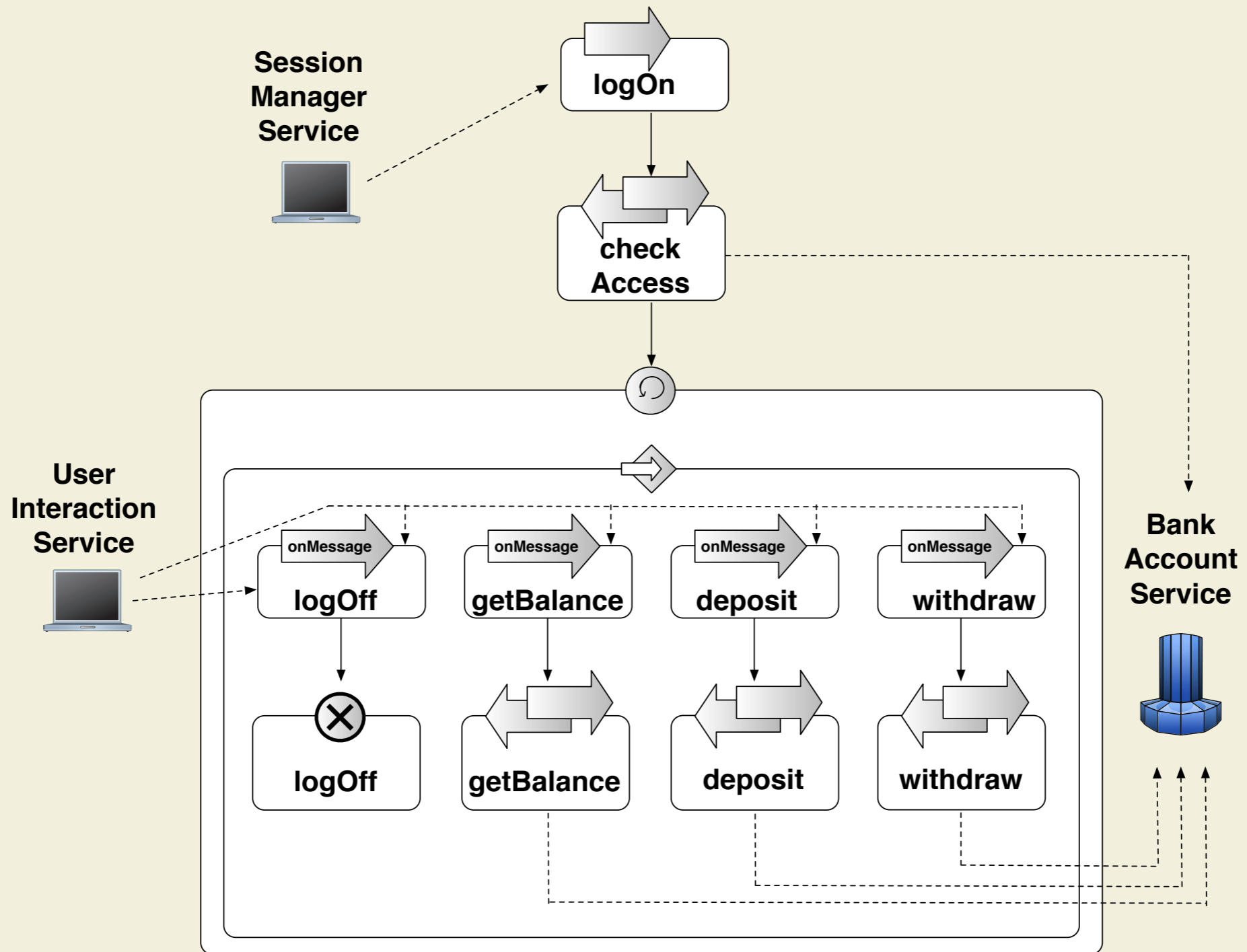
ρ (Counting Modality Formula)

$$\rho \left(\mathbf{c}_{\bowtie n}^k(\phi) \right) \equiv X(c_\phi) - Y^{k-1}(c_\phi) \bowtie n$$

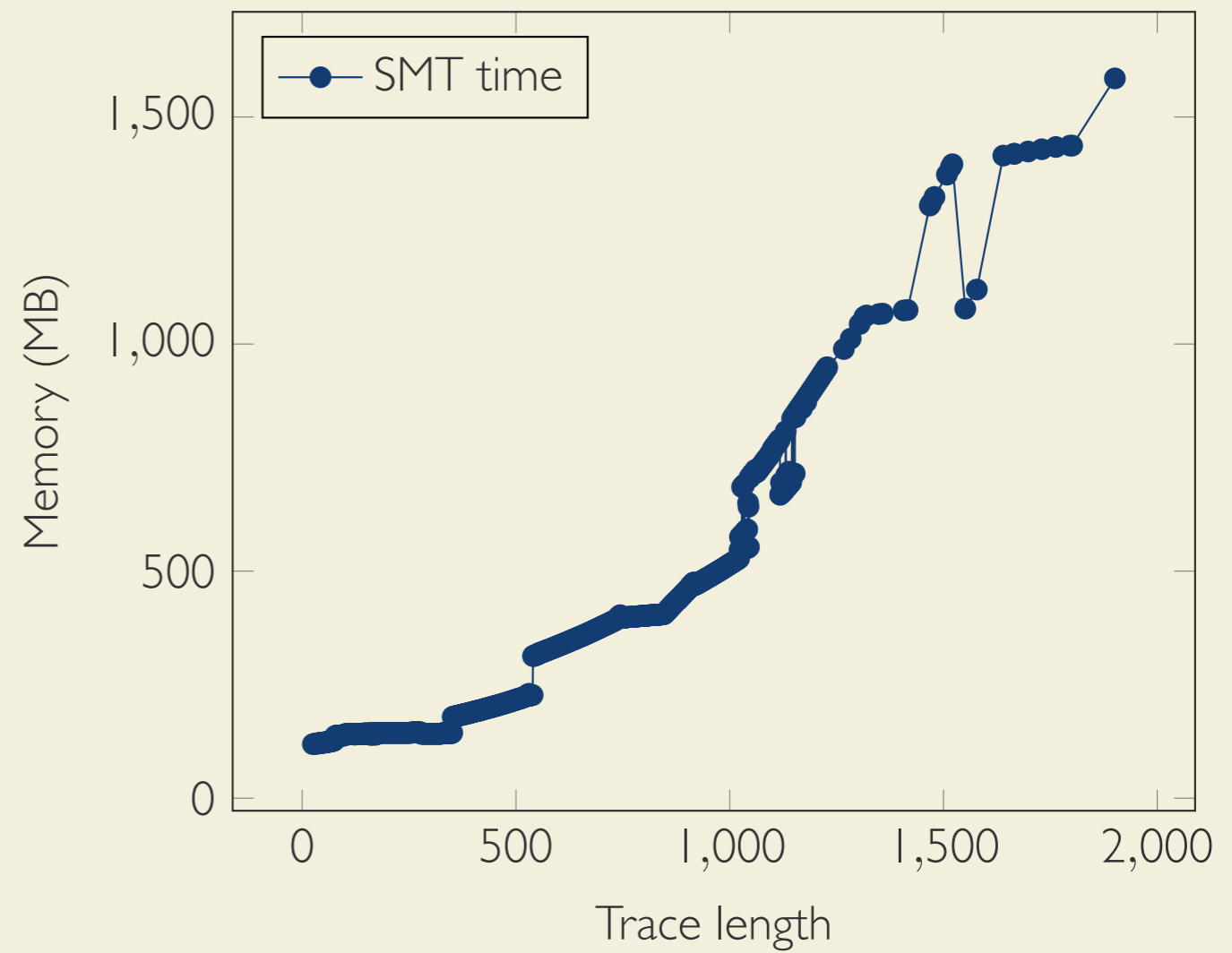
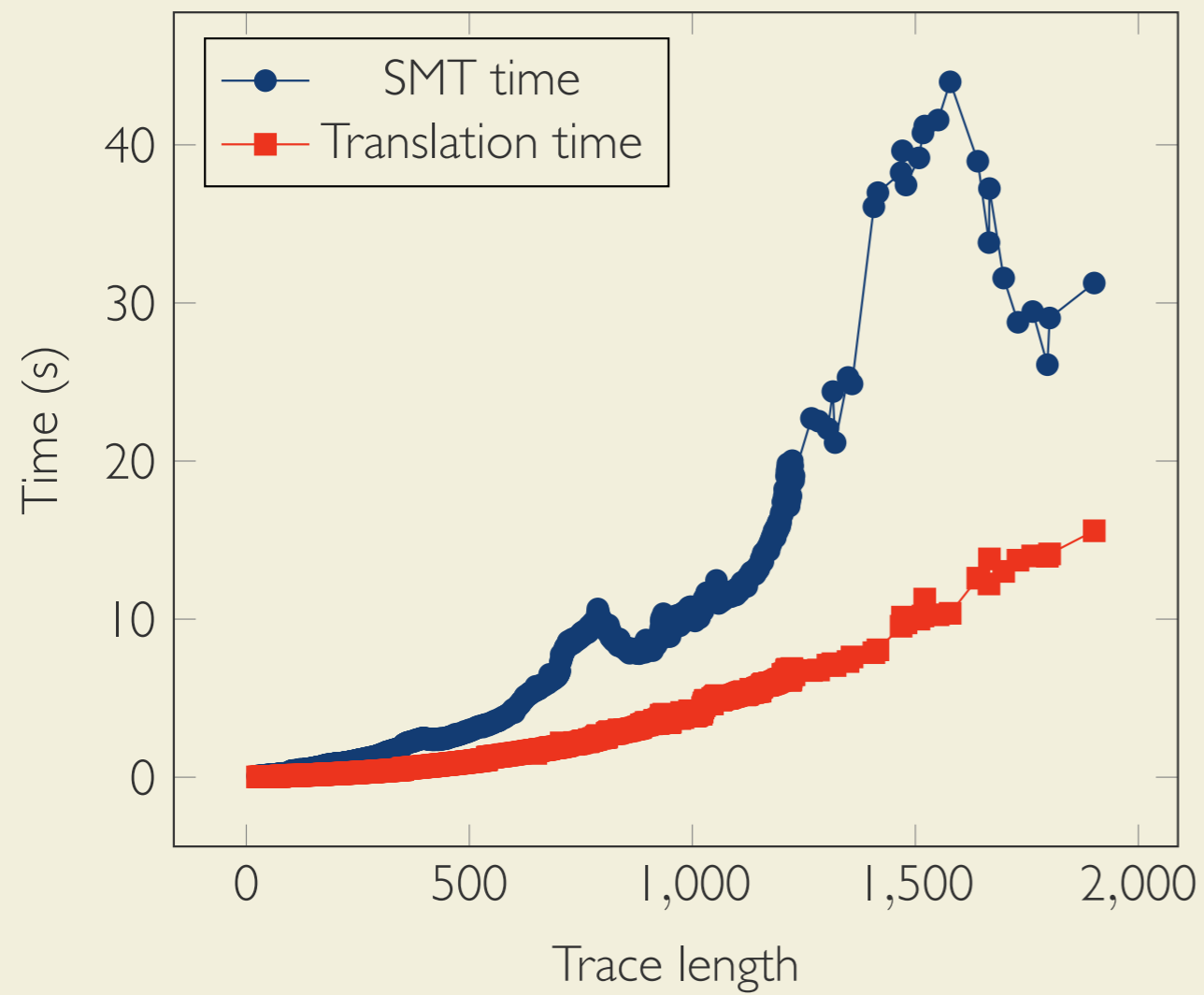
Evaluation

- **RQ1:** Scalability with respect to the various parameters
- **RQ2:** CLTLB(D) vs QF-EUFIDL Encoding
- **RQ3:** CLTLB(D) vs LTL Encoding
- **RQ4:** Application to a Realistic example

Evaluation



RQI: Scalability



RQ2: CLTLB(D) vs QF-EUFIDL

SMT-based Checking of SOLOIST over Sparse Traces

Marcello M. Bersani¹, Domenico Bianculli², Carlo Ghezzi¹, Srđan Krstić¹, and
Pierluigi San Pietro¹

¹ DEEP-SE group - DEIB - Politecnico di Milano, Italy
{bersani,ghezzi,krstic,sanpietr}@elet.polimi.it

² SnT Centre - University of Luxembourg, Luxembourg
domenico.bianculli@uni.lu

Abstract. SMT solvers have been recently applied to bounded model checking and satisfiability checking of metric temporal logic. In this paper we consider SOLOIST, an extension of metric temporal logic with aggregate temporal modalities; it has been defined based on a field study on the use of specification patterns in the context of the provisioning of service-based applications. We apply bounded satisfiability checking to perform trace checking of service execution traces against requirements expressed in SOLOIST. In particular, we focus on sparse traces, i.e., traces in which the number of time instants when events occur is very low with respect to the length of the trace.

The main contribution of this paper is an encoding of SOLOIST formulae into formulae of the theory of quantifier-free integer difference logic with uninterpreted function and predicate symbols. This encoding paves the way for efficient checking of SOLOIST formulae over sparse traces using an SMT-based verification toolkit. We report on the evaluation of the proposed encoding, commenting on its scalability and its effectiveness.

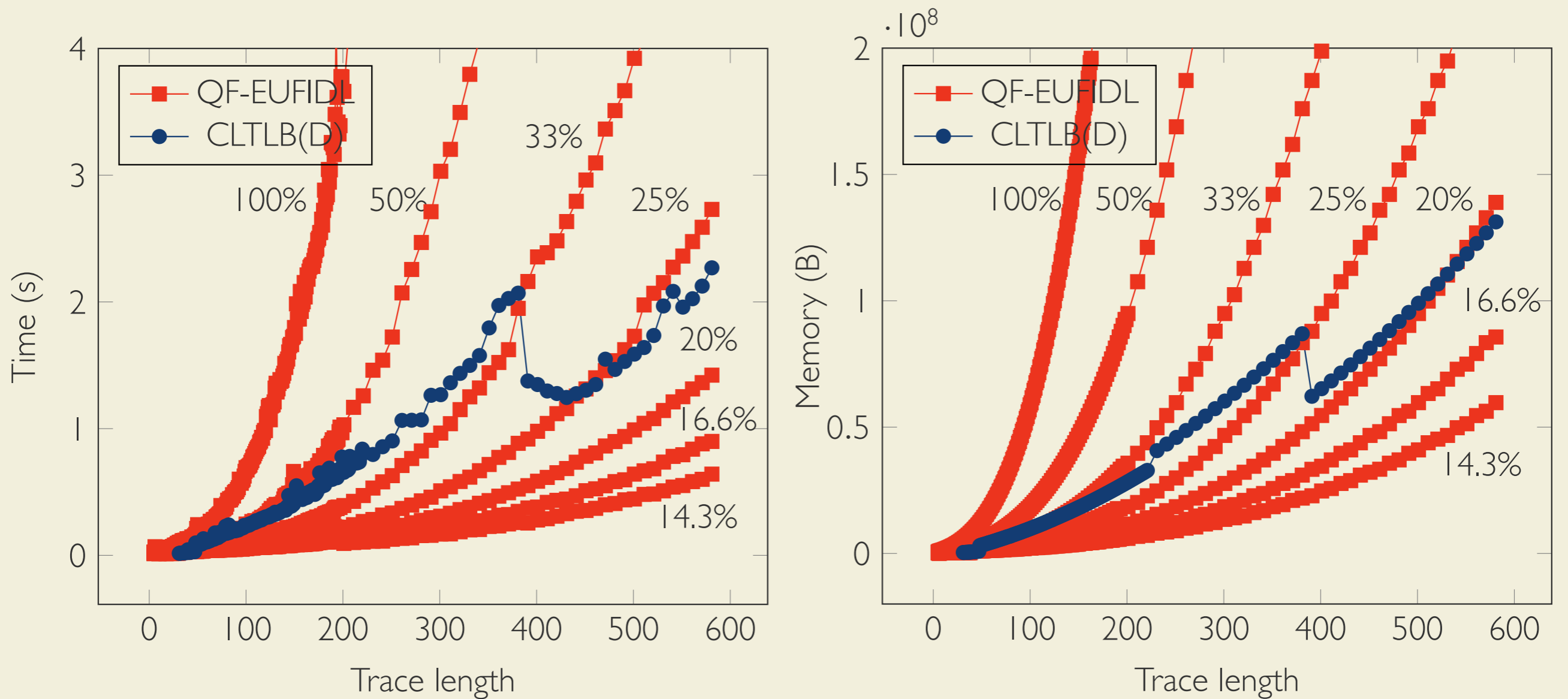
1 Introduction

Bounded satisfiability checking [23] (BSC) is a verification technique that complements bounded model checking [9] (BMC): instead of a customary operational model (e.g., a state-transition system) used in BMC, BSC supports the analysis of a *descriptive model*, denoted by a set of temporal logic formulae. With BSC, verification tasks become suitable instances of the satisfiability problem for quite large formulae (written in a certain logic), which comprehend the model of the system to analyze as well as the requirement(s) to verify. BSC has been successfully applied in the context of metric temporal logics and implemented in ZOT [23], a verification toolset based on SAT- and SMT-solvers, developed within our group.

In this paper we apply BSC to trace checking for the language SOLOIST (*Specification Language for service compositions in Interactions*) [8], a metric temporal logic with new, additional temporal modalities that support aggregate operations on events occurring in a given time window. SOLOIST has been defined based on the results of a field study [7] on the type of property specification patterns used to express requirements in the context of service-based applications. The study—performed by some of

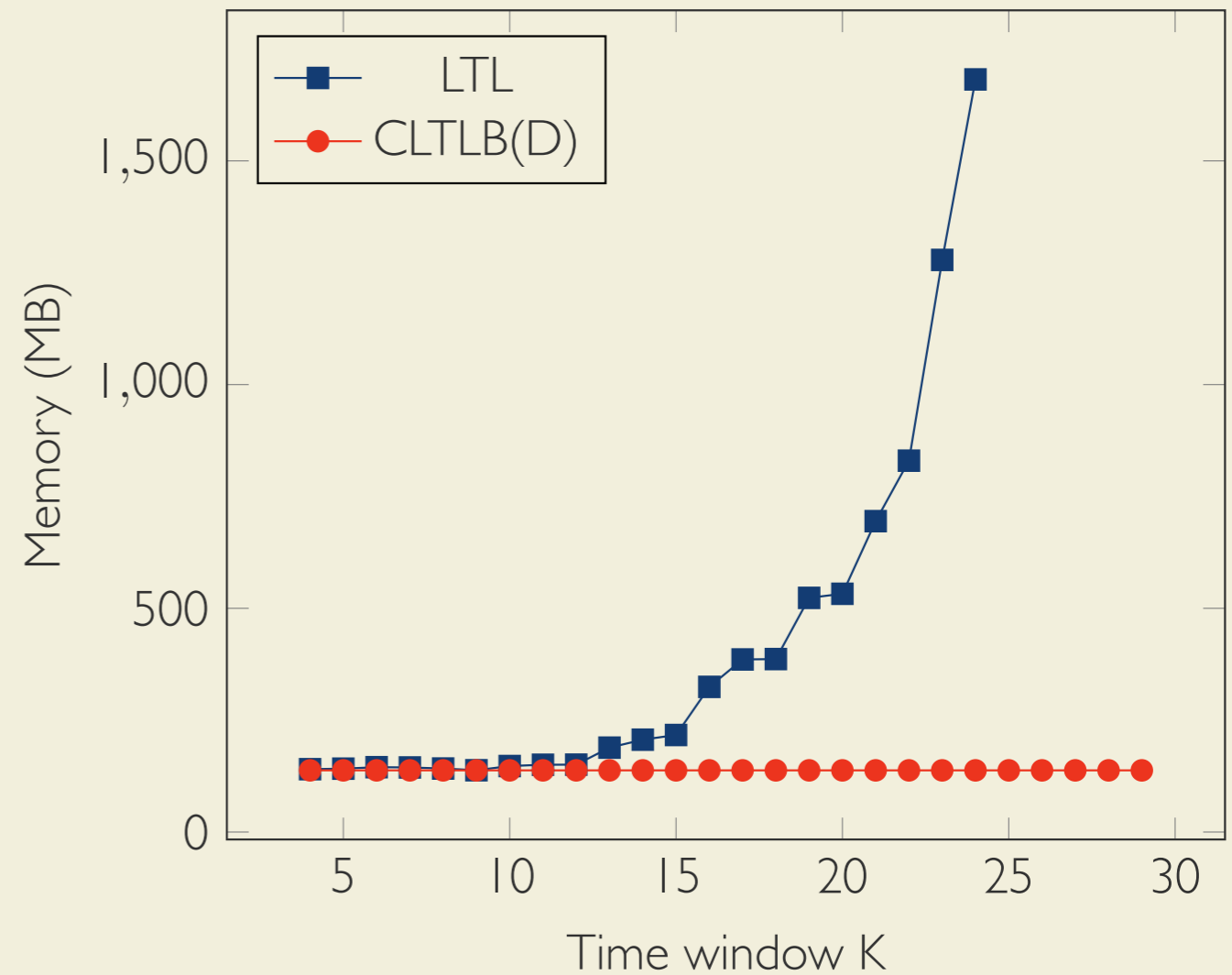
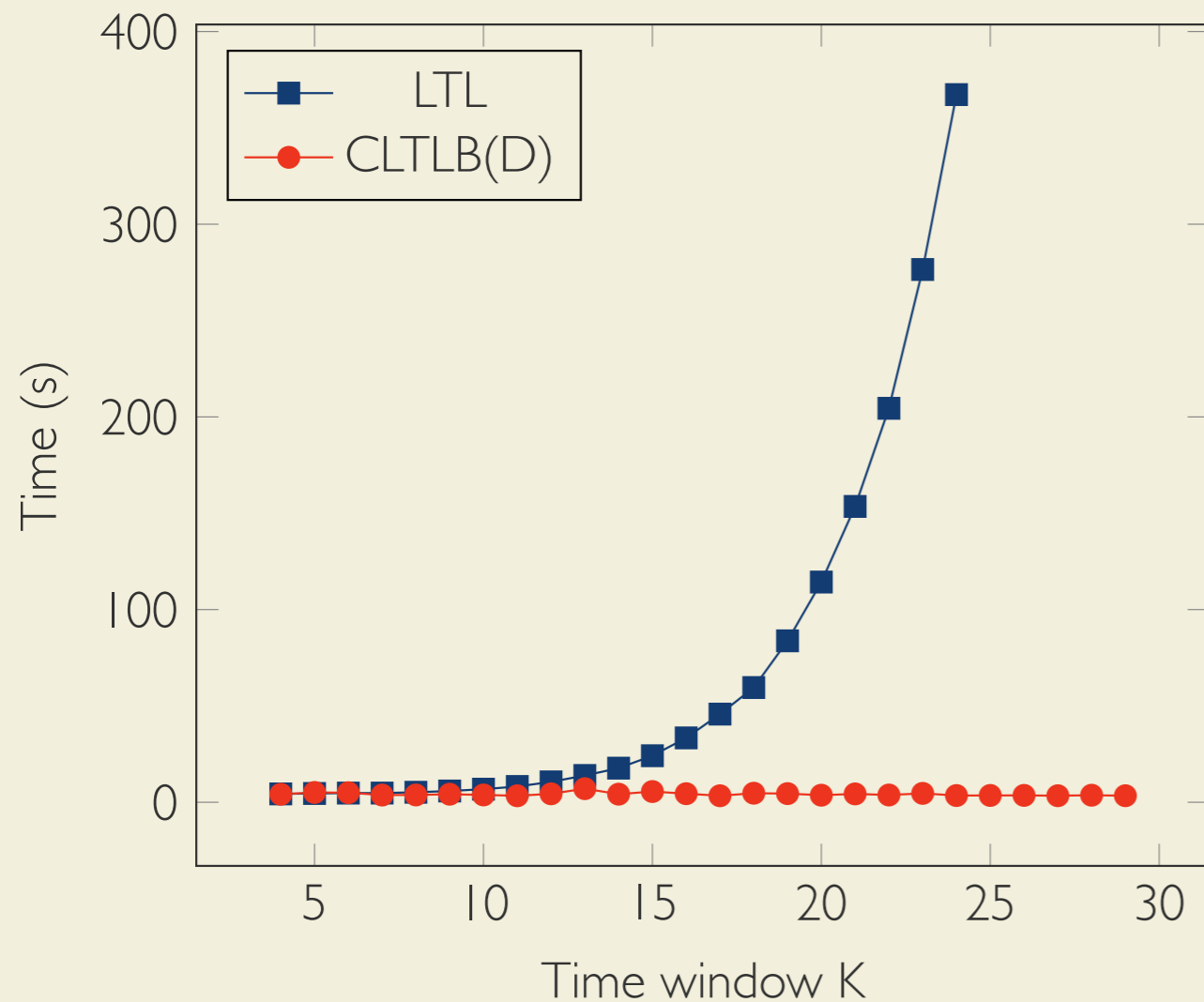
FASE 2014

RQ2: CLTLB(D) vs QF-EUFIDL



Marcello Maria Bersani, Domenico Bianculli, Carlo Ghezzi, Srđan Krstić, and Pierluigi San Pietro: SMT-based checking of SOLOIST over sparse traces. In Proc of FASE2014

RQ3: CLTLB(D) vs LTL



RQ4: Application

QP1: $G(\text{logOff} \rightarrow \mathfrak{C}_{\leq 3}^{600}(\text{withdraw}_{\text{end}}))$

“The number of withdrawal operations performed within 10 minutes before customer logs off is less than or equal to 3”

QP2: $G(\mathfrak{D}_{< 5}^{900} \{(\text{checkAccess}_{\text{start}}, \text{checkAccess}_{\text{end}})\})$

“The average response time of operation checkAccess provided by the BankAccount service is always less than 5 seconds within any 15 minute time window”

QP3: $G(\text{logOff} \rightarrow \mathfrak{M}_{\leq 2}^{600, 60}(\text{getBalance}_{\text{end}}))$

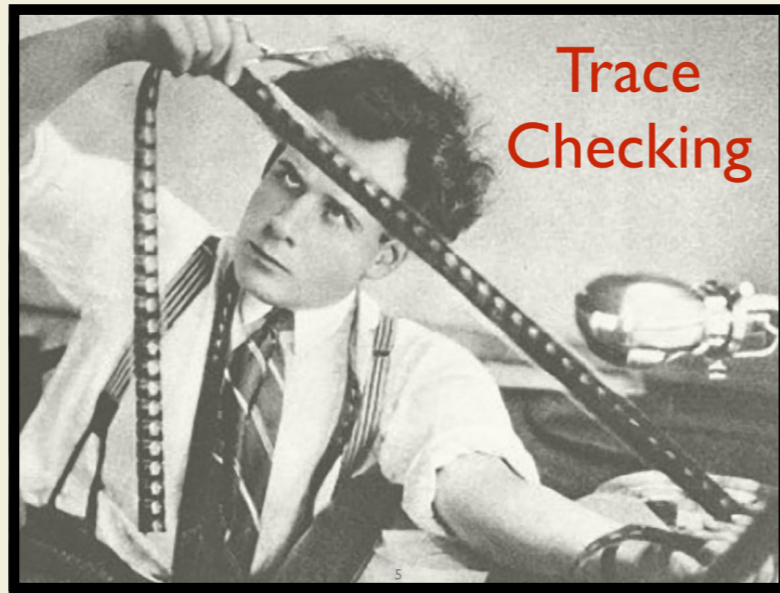
“The maximum number of balance inquiries is restricted to at most 2 per minute within 10 minutes before customer session ends”

RQ4: Application

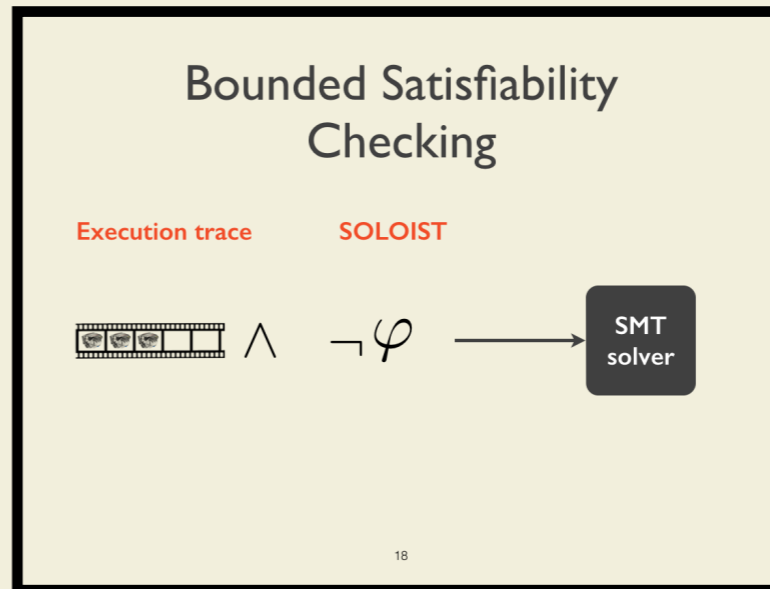
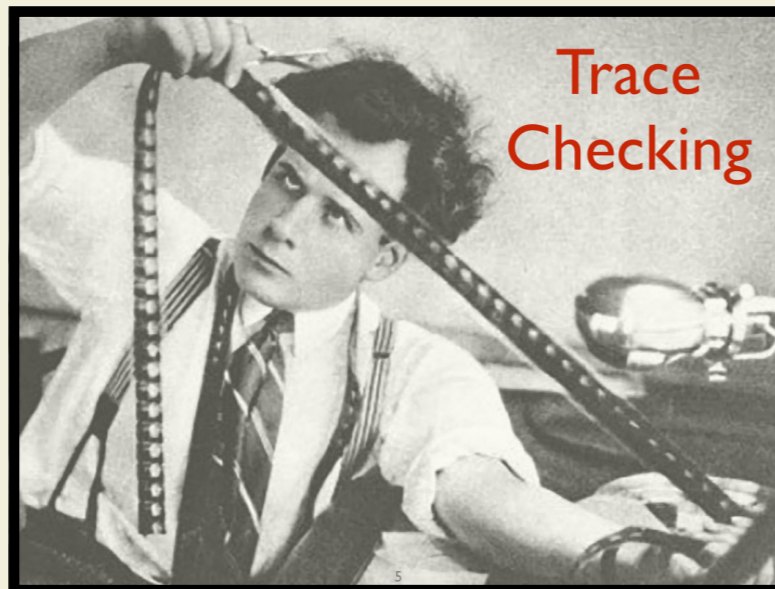
Property	Outcome	ZOT time (s)	SMT time (s)	Total time (s)	Memory (MB)
(mean over 30K traces)					
(QP1)	true	10.97	19.70	30.27	931.0
(QP2)	false	20.77	107.30	125.50	1261.7
(QP3)	true	10.56	31.89	42.08	793.0
(\neg QP1)	false	11.59	28.00	38.94	932.0
(\neg QP2)	true	20.74	324.8	343.0	1210.8
(\neg QP3)	false	11.10	31.37	41.97	942.0

Offline Trace Checking of Quantitative Properties of Service-Based Applications

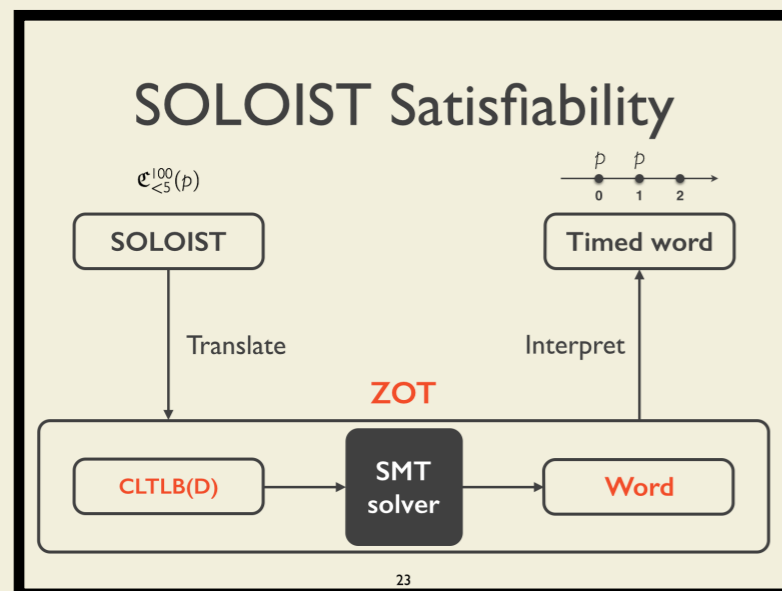
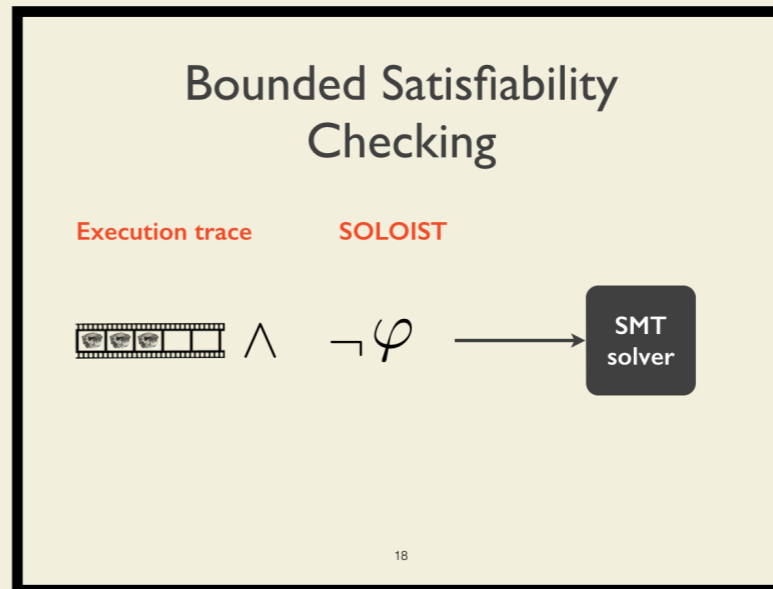
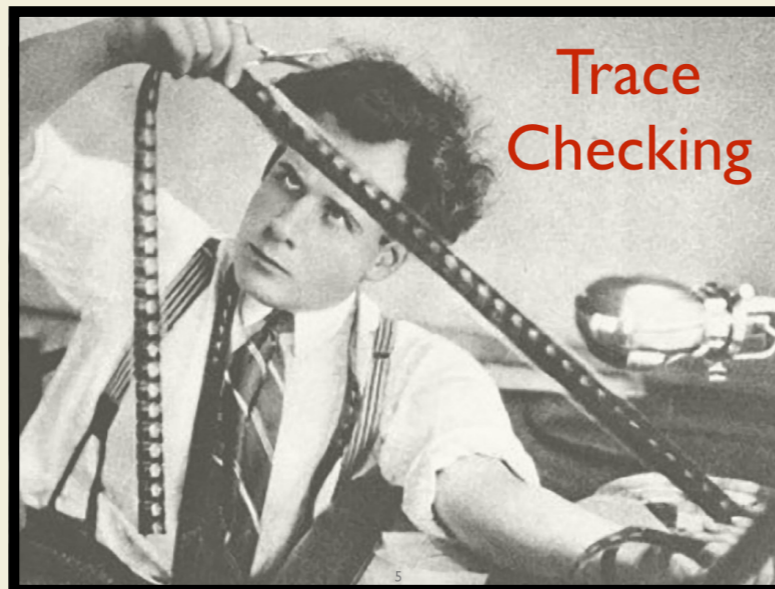
Offline Trace Checking of Quantitative Properties of Service-Based Applications



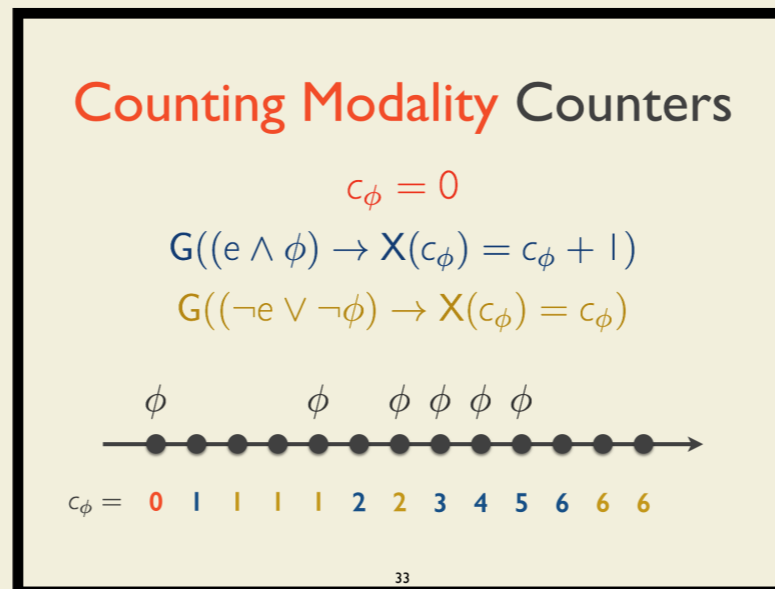
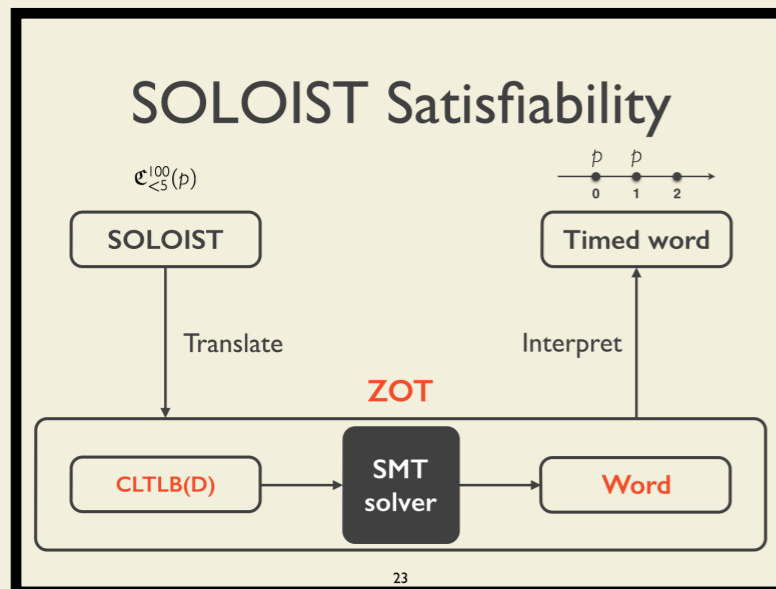
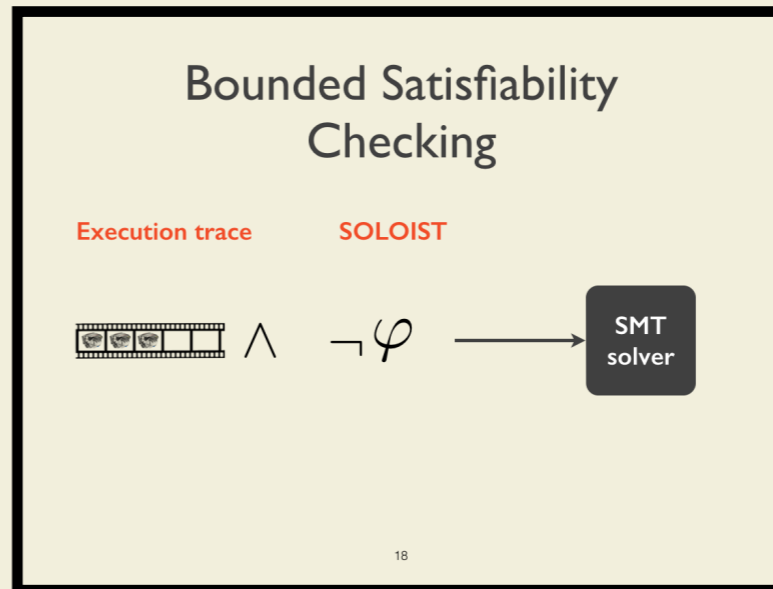
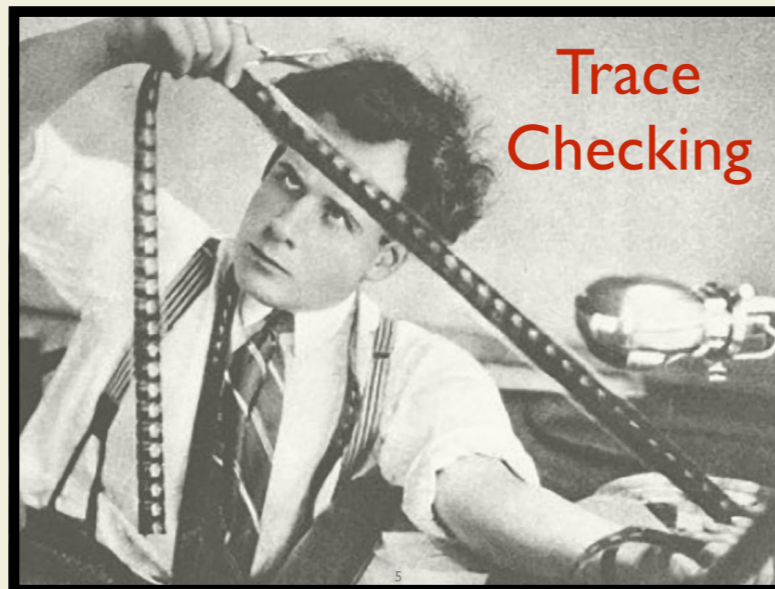
Offline Trace Checking of Quantitative Properties of Service-Based Applications



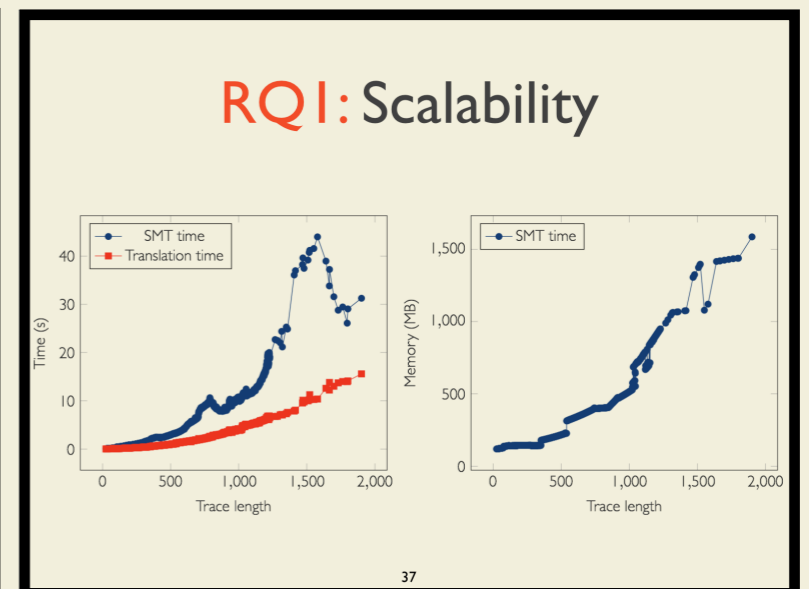
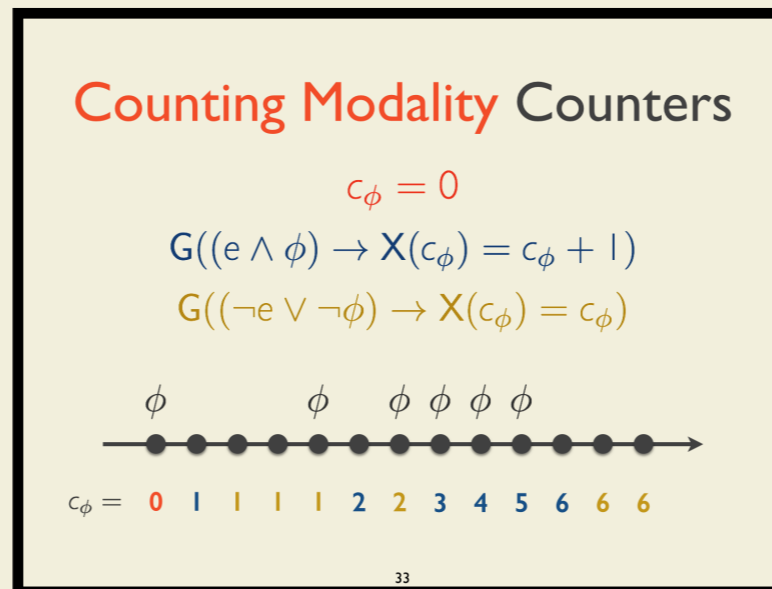
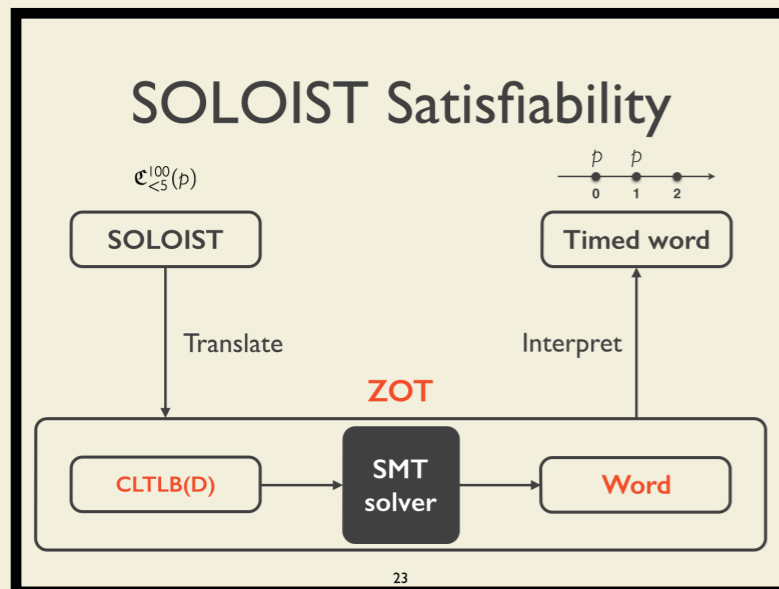
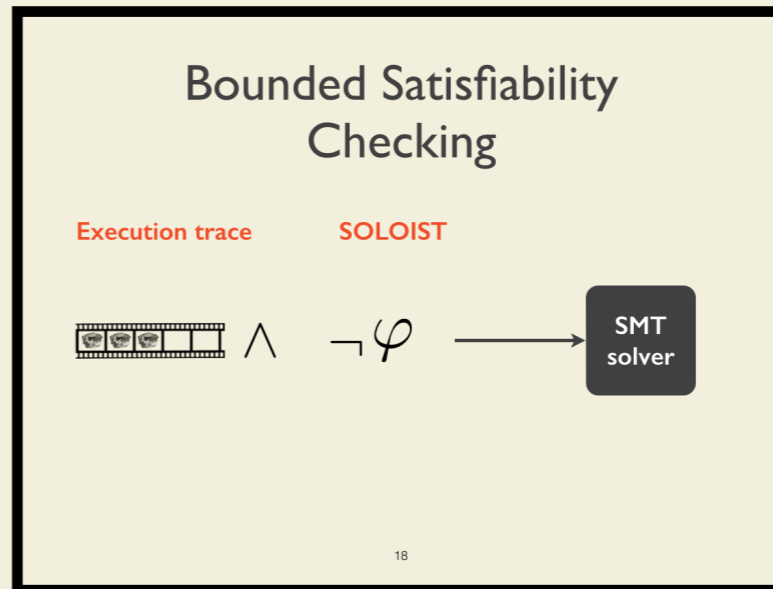
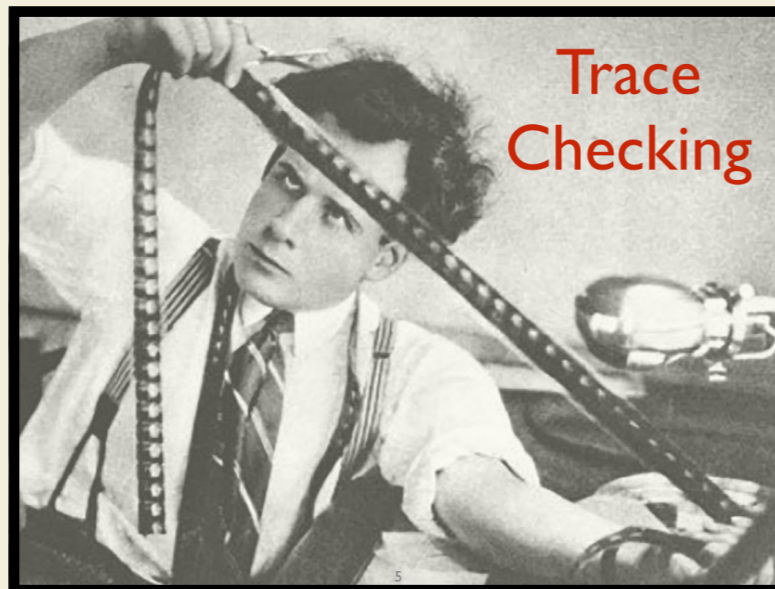
Offline Trace Checking of Quantitative Properties of Service-Based Applications



Offline Trace Checking of Quantitative Properties of Service-Based Applications



Offline Trace Checking of Quantitative Properties of Service-Based Applications



Future Work

- Adaptation to run-time verification
- Comparison to a distributed offline trace checking algorithm based on MapReduce
- Application to a more extensive case studies

Offline Trace Checking of Quantitative Properties of Service-Based Applications

Srđan Krstić

with

Domenico Bianculli, Carlo Ghezzi and Pierluigi San Pietro

