

# SMT-based Checking of SOLOIST over Sparse Traces

Srđan Krstić

with

Marcello Bersani, Domenico Bianculli, Carlo Ghezzi and Pierluigi San Pietro



# SMT-based Checking of SOLOIST over Sparse Traces

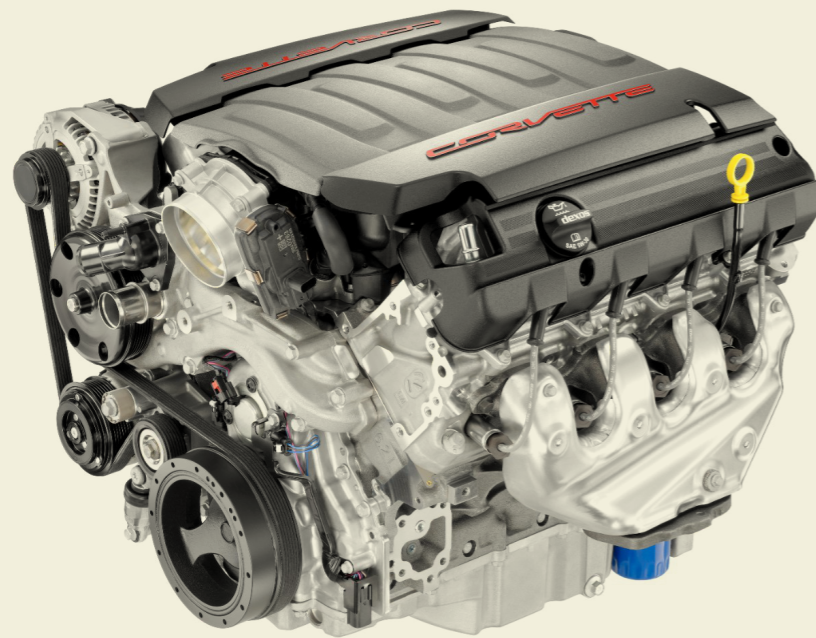
Srđan Krstić

with

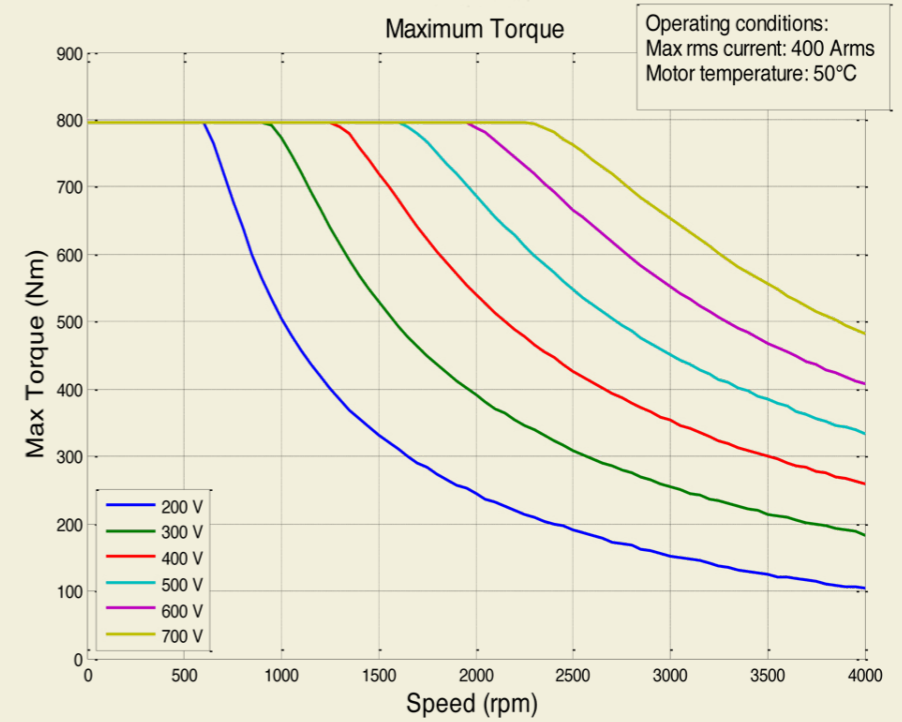
Marcello Bersani, Domenico Bianculli, Carlo Ghezzi and Pierluigi San Pietro



# System



# Specification

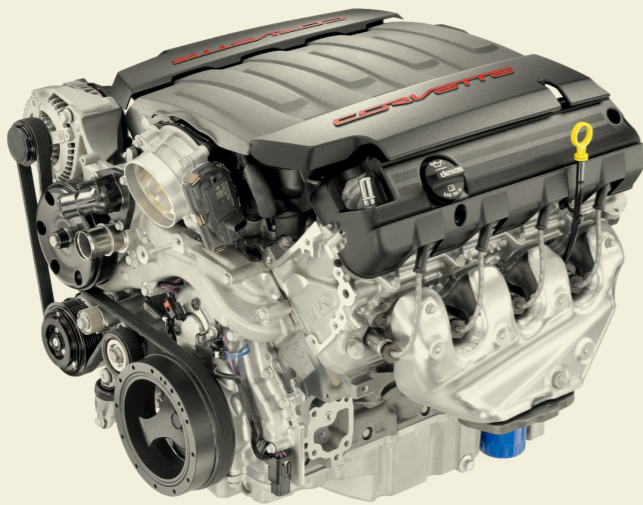


# Execution Traces



# Execution Traces

**System**

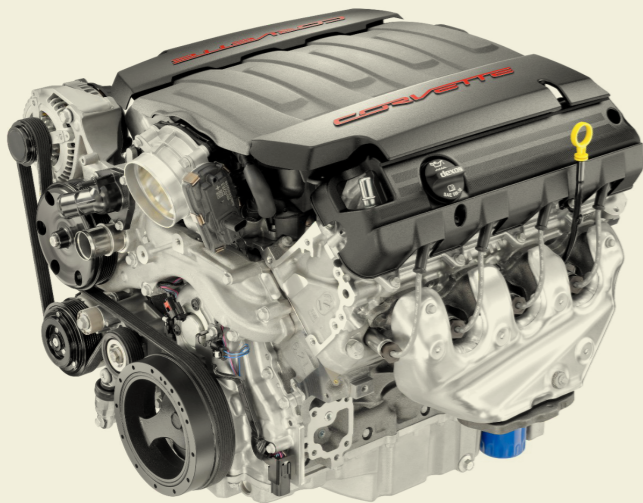


# Execution Traces



Monitor

System

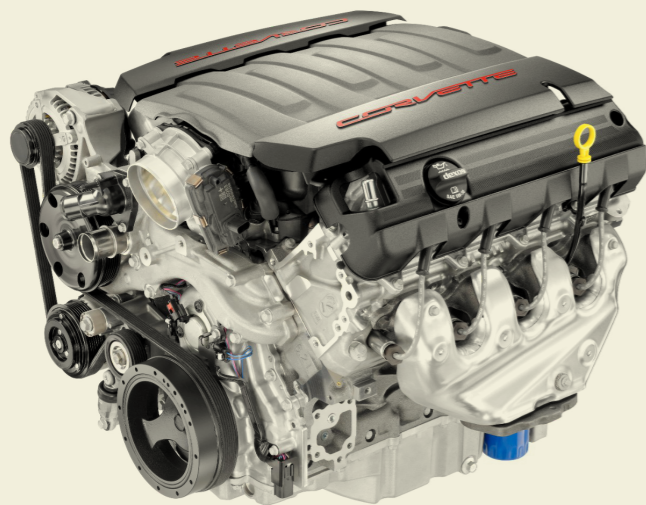


# Execution Traces

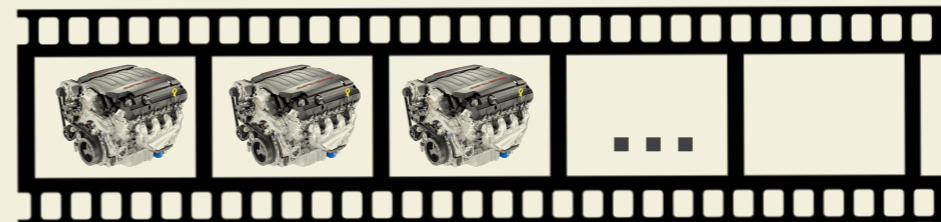


Monitor

System

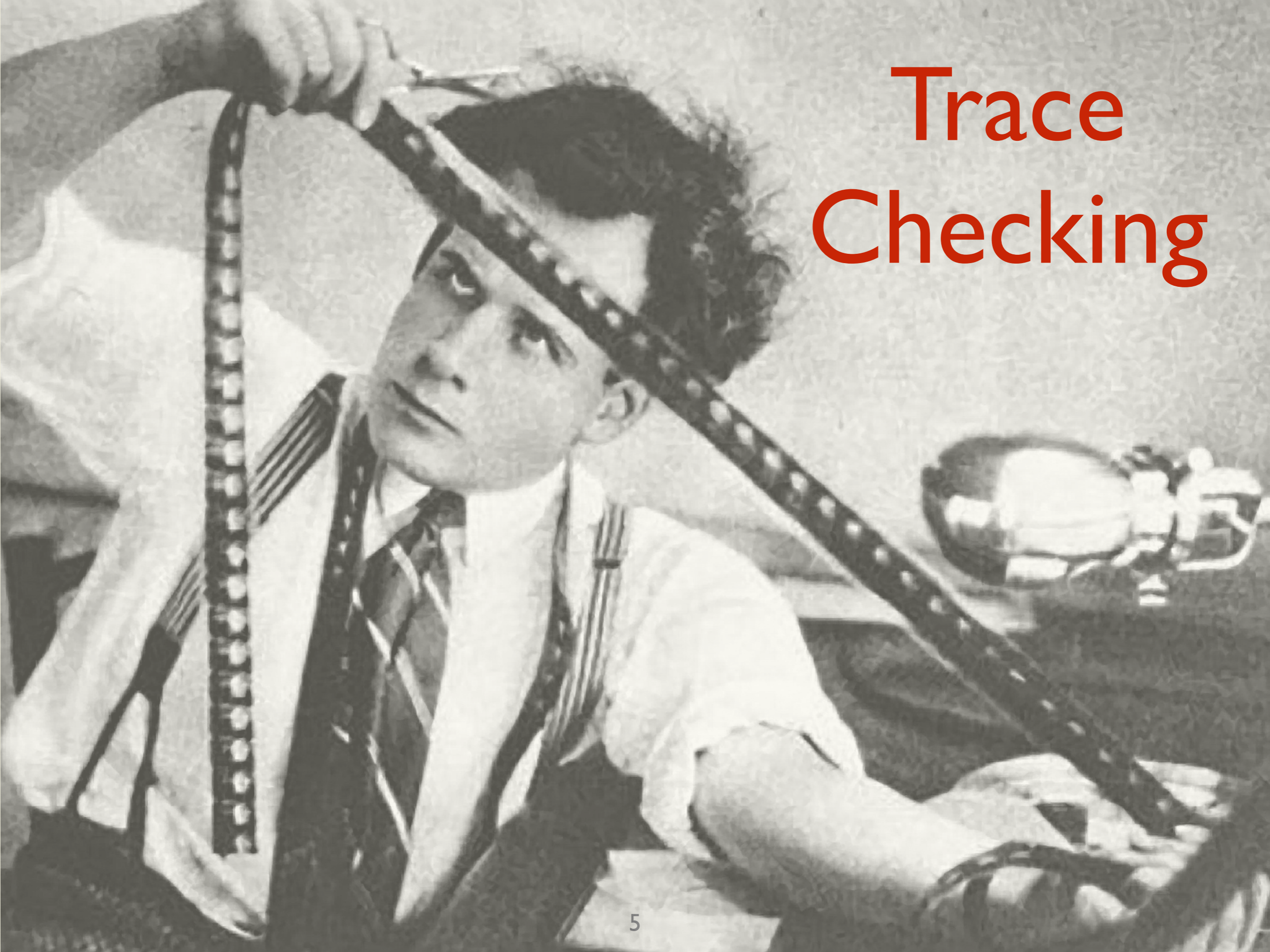


Execution trace





# Trace Checking

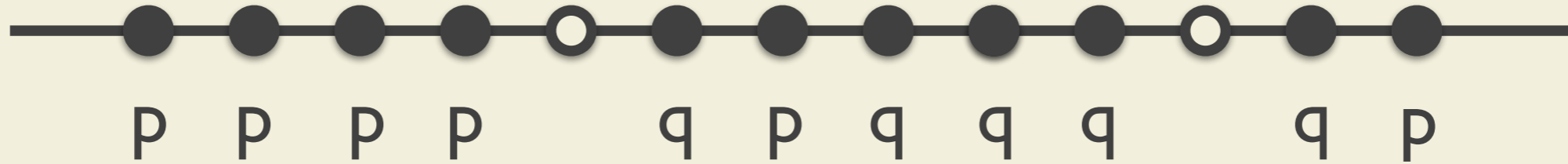




# Sparse Traces

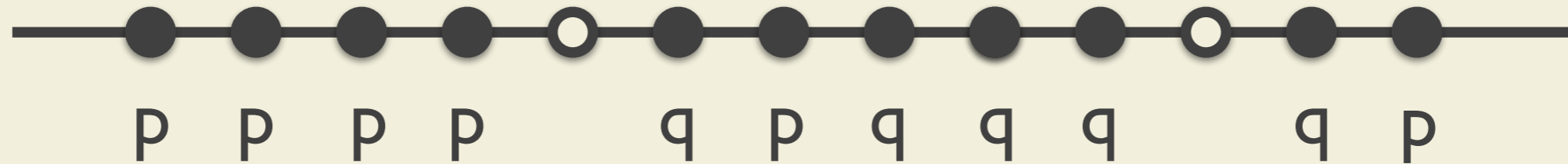
# Sparse Traces

Dense

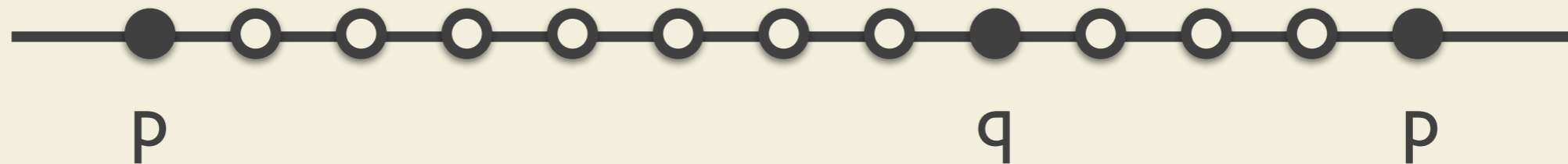


# Sparse Traces

Dense

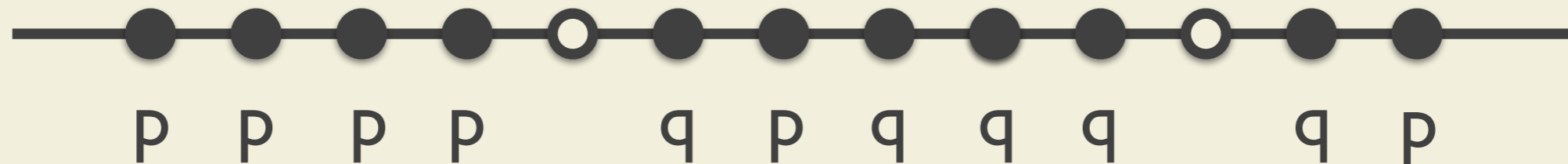


Sparse

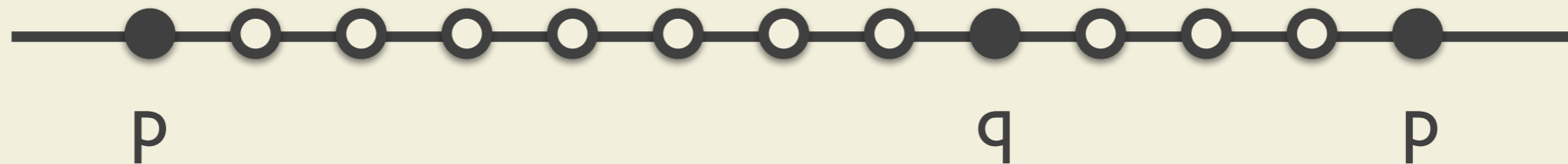


# Sparse Traces

Dense



Sparse



$$\text{sparseness} = \frac{\# \text{ of relevant time instants}}{\# \text{ of time instants}}$$



# Sparse Traces

Business Process Intelligence Challenge 2012

Traces taken from a Dutch Financial Institute

Number of Traces: 13087

**Average Events per Trace: 20.03**

**Average Trace Duration: 8.62 Days**

# Sparse Traces

Business Process Intelligence Challenge 2012

Traces taken from a Dutch Financial Institute

Number of Traces: 13087

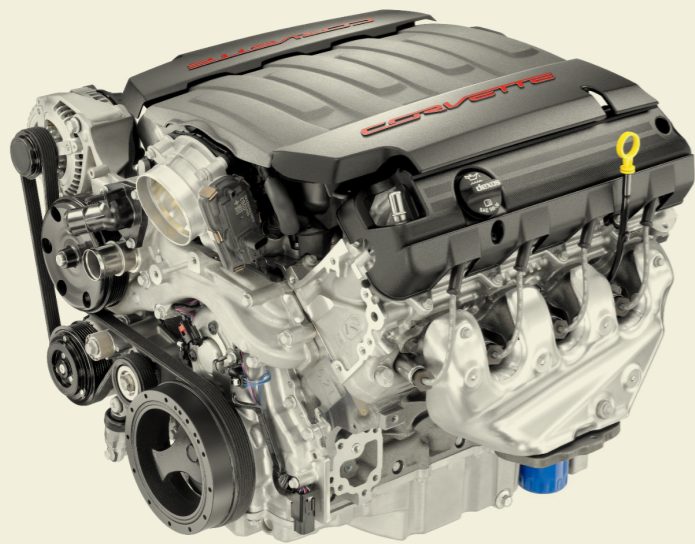
**Average Events per Trace: 20.03**

**Average Trace Duration: 8.62 Days**

**Average sparseness: 0.003%**

# Bounded Satisfiability Checking

System




Specification

$\varphi$

# Bounded Satisfiability Checking

System

Specification

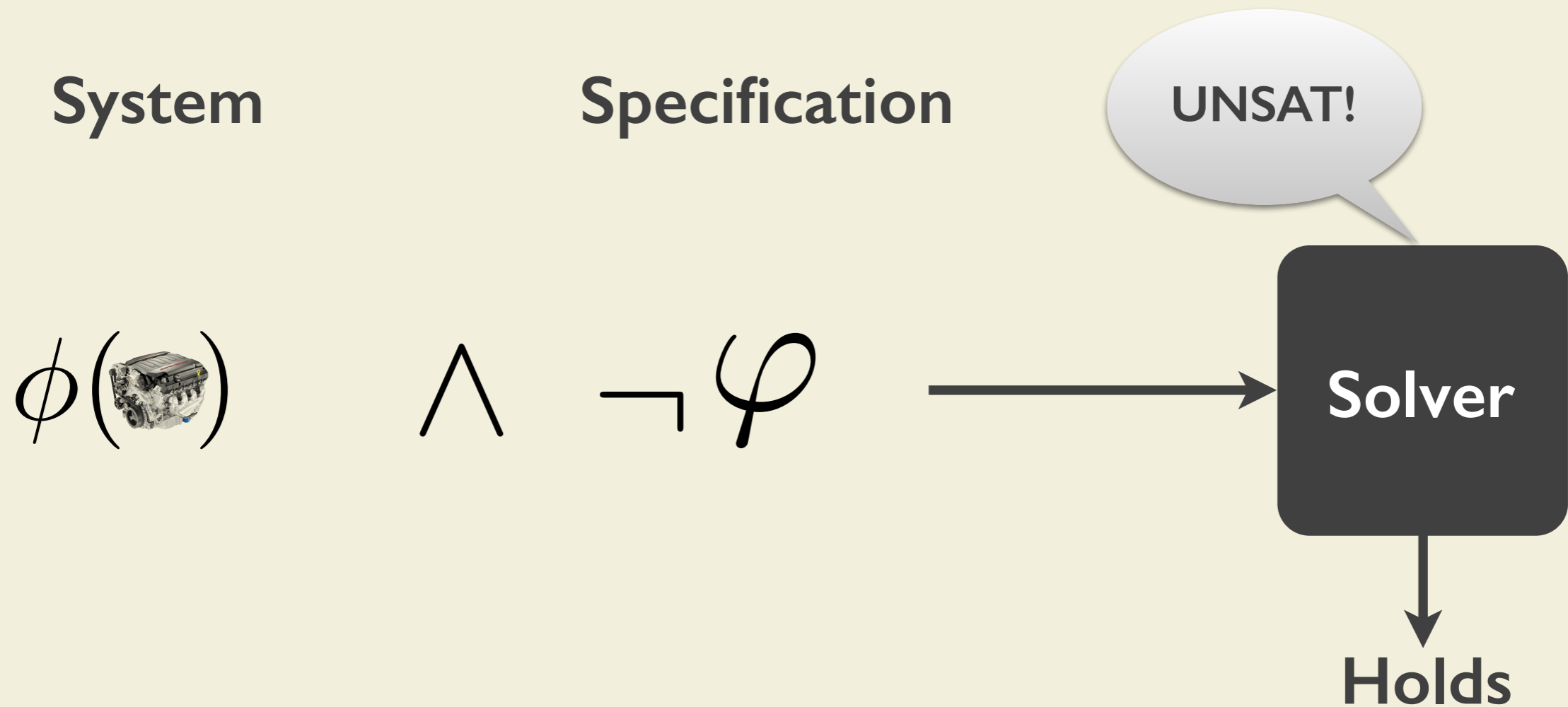
$\phi$  ()

$\wedge \neg \varphi$



Solver

# Bounded Satisfiability Checking

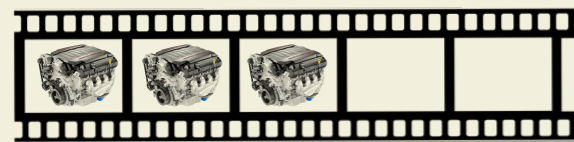




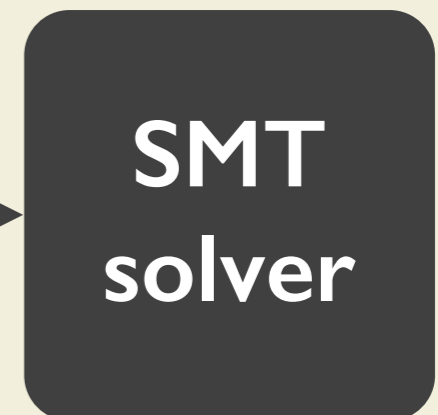
# Bounded Satisfiability Checking

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 Bianculli<sup>1</sup> and Carlo Ghezzi<sup>2</sup> and Pierluigi San Pietro<sup>2</sup>

<sup>1</sup> University of Luxembourg - SnT Centre, Luxembourg  
domenico.bianculli@uni.lu

<sup>2</sup> 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

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

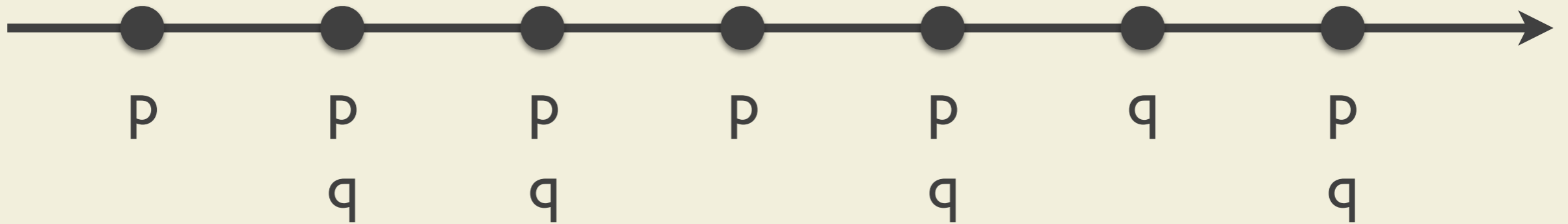
## with aggregating modalities

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



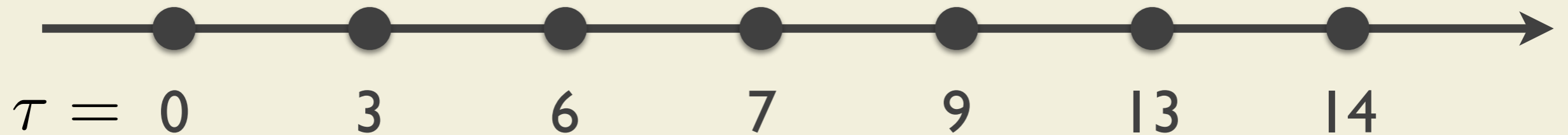
# SOLOIST

Model: a timed  $\omega$ -word

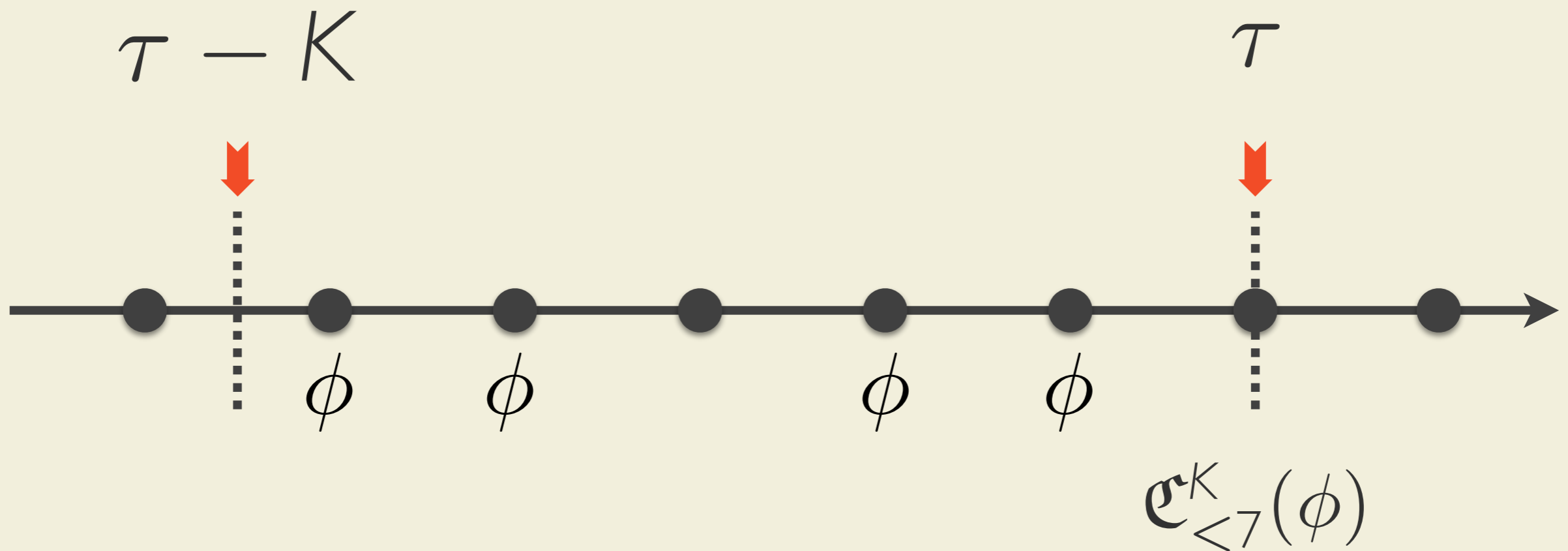


# SOLOIST

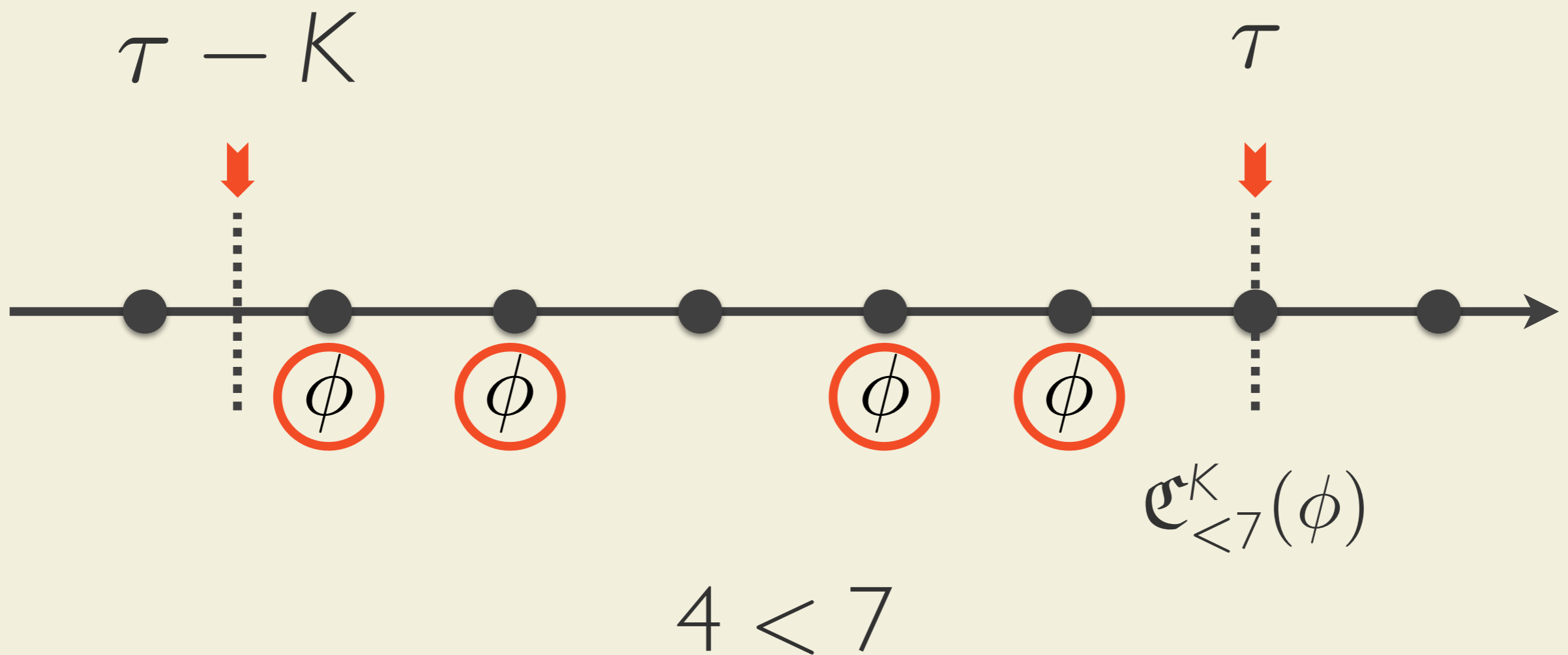
Model: a timed  $\omega$ -word



# Counting Modality



# Counting Modality

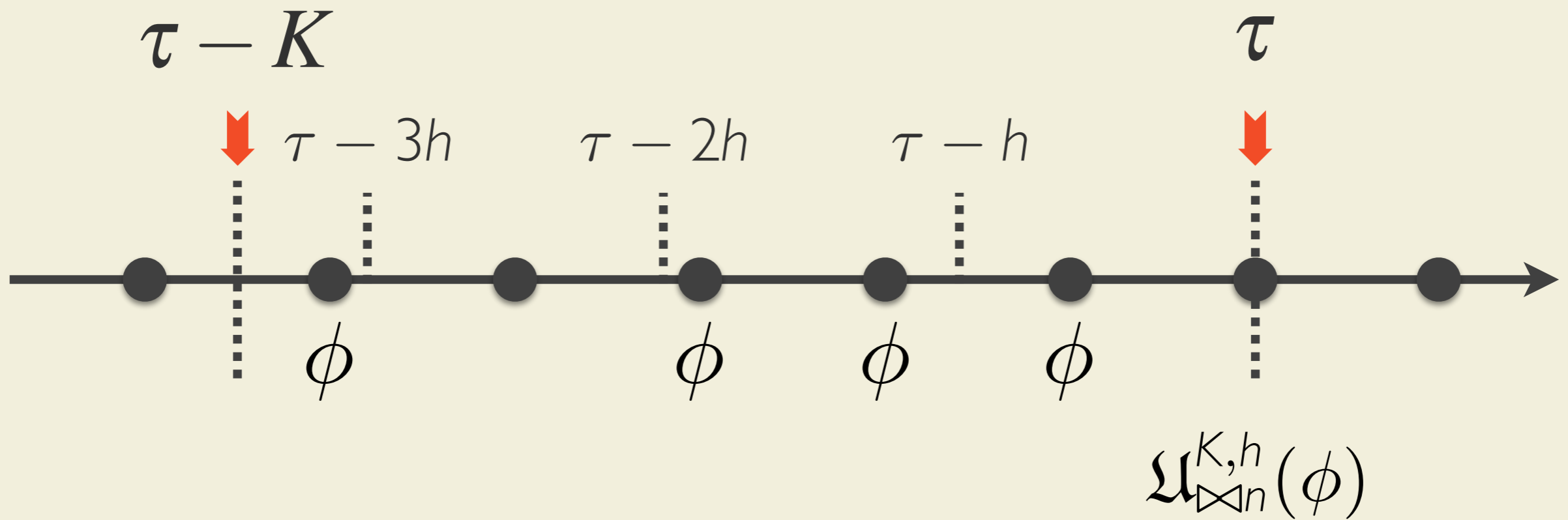


# Counting Modality

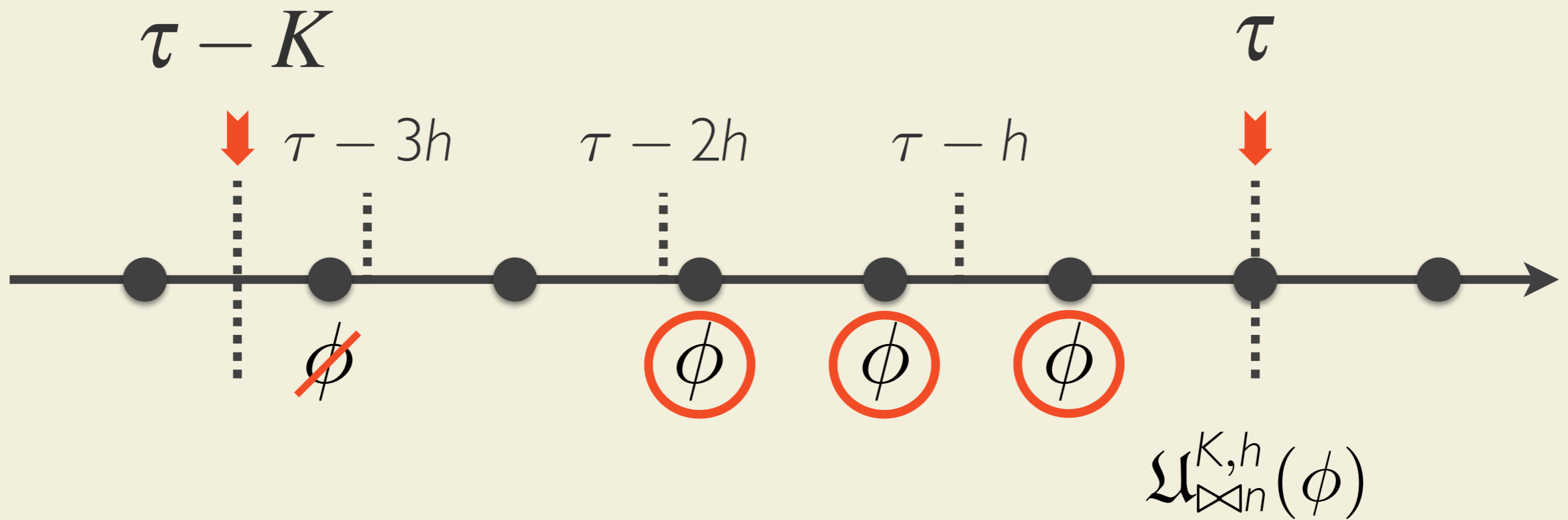
“If operation  $A$  has been executed **exactly 4 times** in the **past 5 minutes**, than operation  $B$  will be executed **within 32 seconds.**”

$$G(\mathcal{C}_{=4}^{300}(A) \rightarrow F_{[0,32]}(B))$$

# SOLOIST



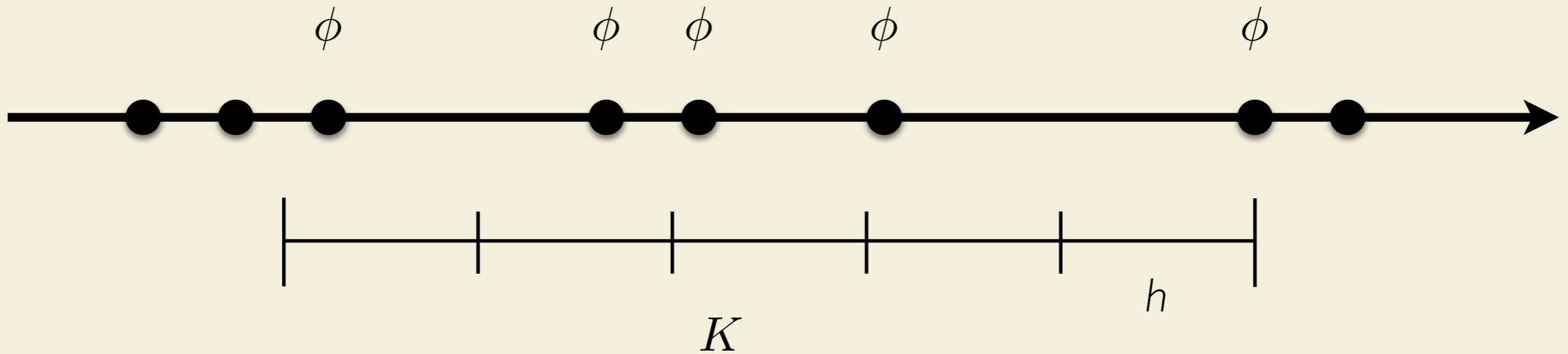
# SOLOIST



$$\frac{0+2+1}{3} \boxtimes n$$

# Average modality

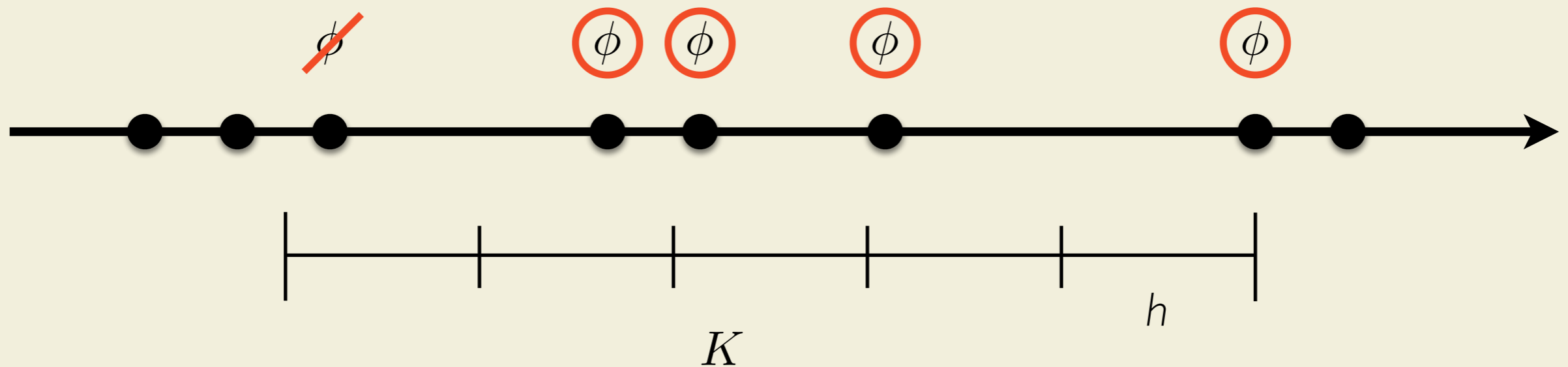
$$\mathcal{U}_{\Delta n}^{K,h}(\phi)$$





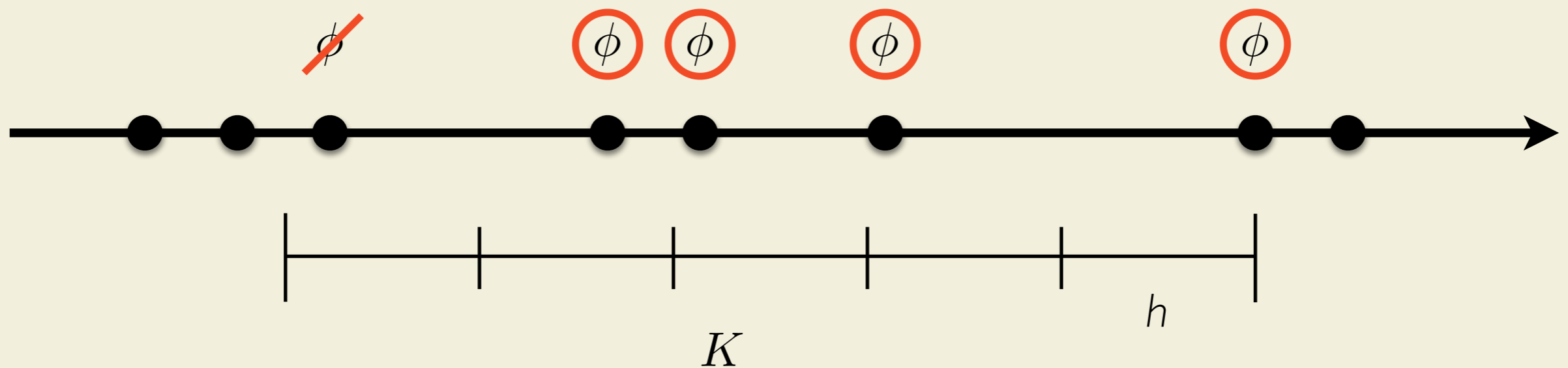
# Average modality

$$\mathcal{U}_{\bowtie n}^{K,h}(\phi) \equiv \frac{1+1+1+1}{4} \bowtie n$$



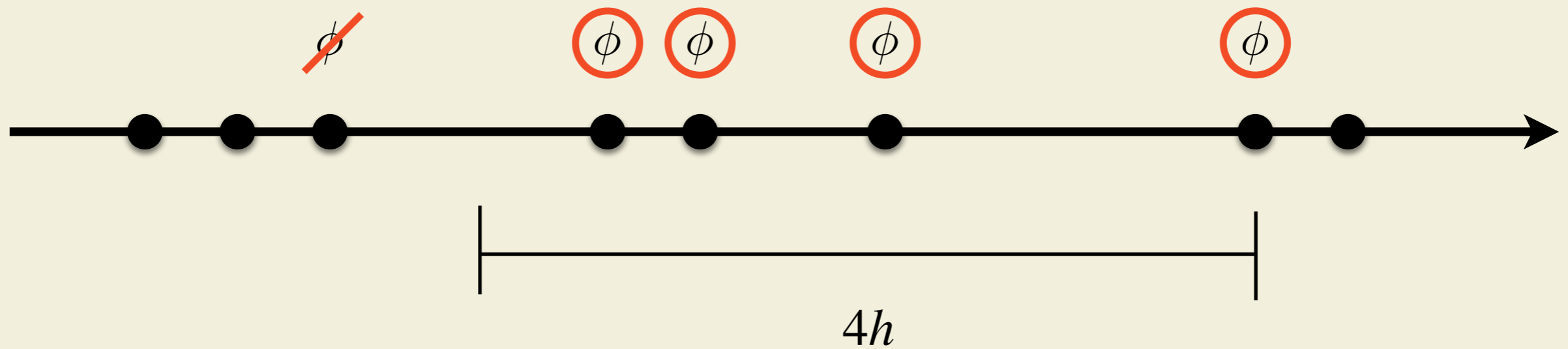
# Average modality

$$\mathcal{U}_{\bowtie n}^{K,h}(\phi) \equiv \frac{1+1+1+1}{4} \bowtie n \equiv 1+1+1+1 \bowtie 4 \cdot n$$



# Average modality

$$\mathcal{U}_{\boxtimes n}^{K,h}(\phi) \equiv \frac{1+1+1+1}{4} \boxtimes n \equiv 1+1+1+1 \boxtimes 4 \cdot n \equiv \mathcal{E}_{\boxtimes 4n}^{4h,h}(\phi)$$



# Average Modality

“When operation  $A$  is executed, **the average number** of executions, in an **interval of 1 minute**, of operation  $B$  during the **past 12 minutes** should be **less than 4.**”

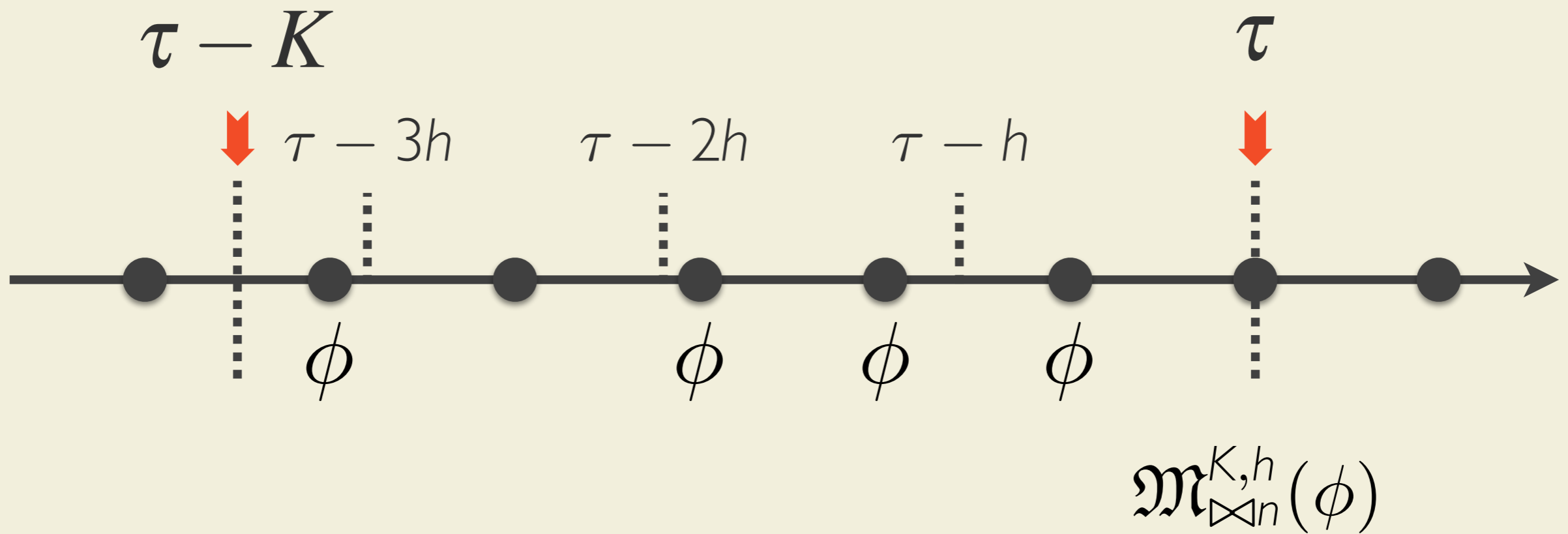
$$G(A \rightarrow \mathcal{U}_{<4}^{720,60}(B))$$

# Maximum Modality

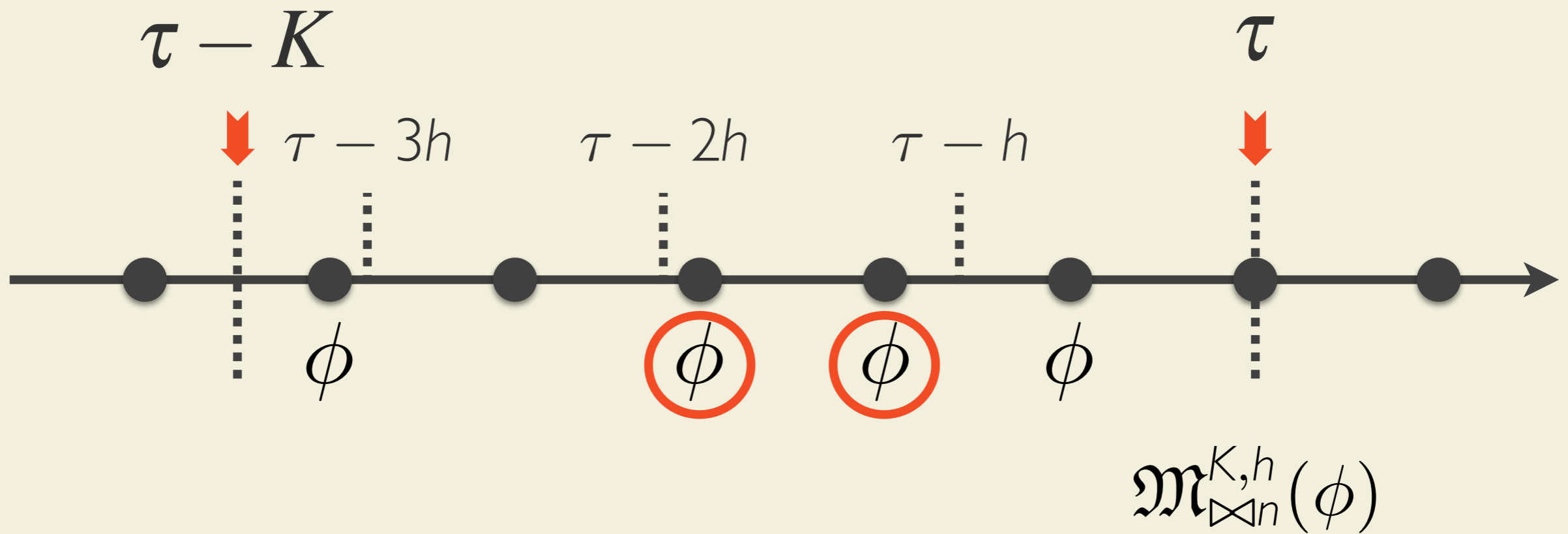
“When operation  $A$  is executed, the **maximum number** of executions, in an **interval of 1 minute**, of operation  $B$  during the **past 12 minutes** should be **greater than 5**”

$$G(A \rightarrow \mathfrak{M}_{>5}^{720,60}(B))$$

# SOLOIST



# SOLOIST



$$2 \bowtie n$$

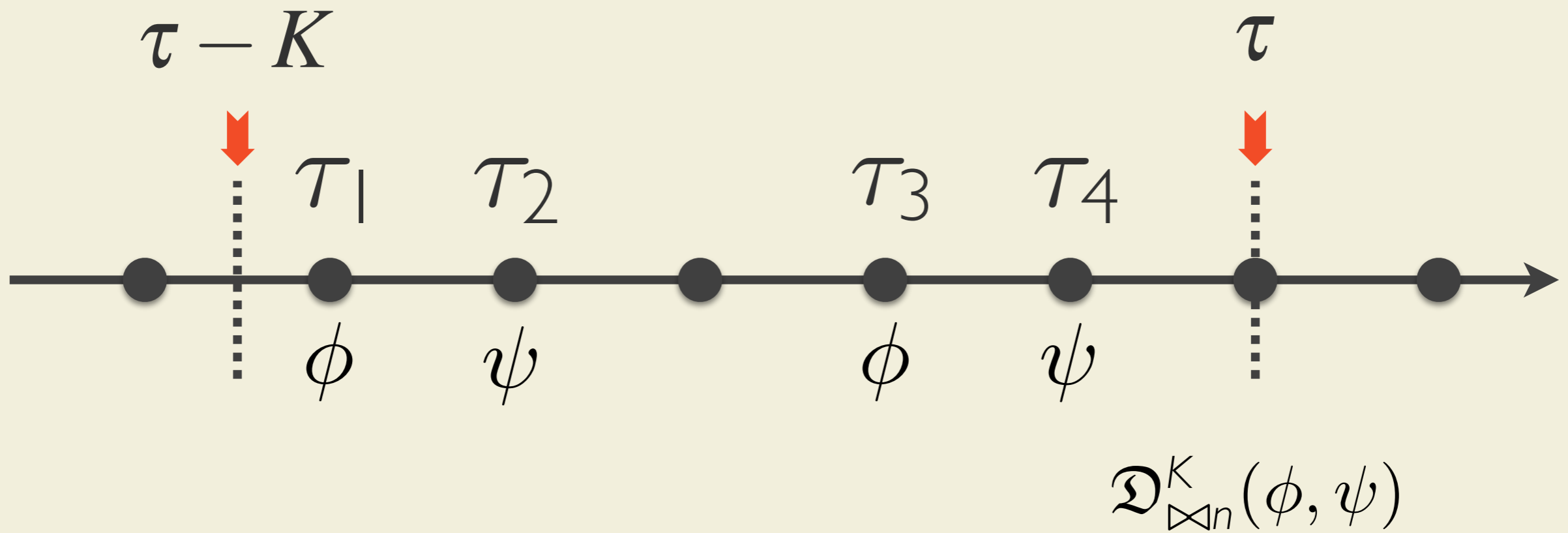
# Average Distance Modality

“When operation  $A$  is executed, the **average response time** of all the executions of operation  $C$  in the **past 12 minutes** should be **less than 3 seconds.**”

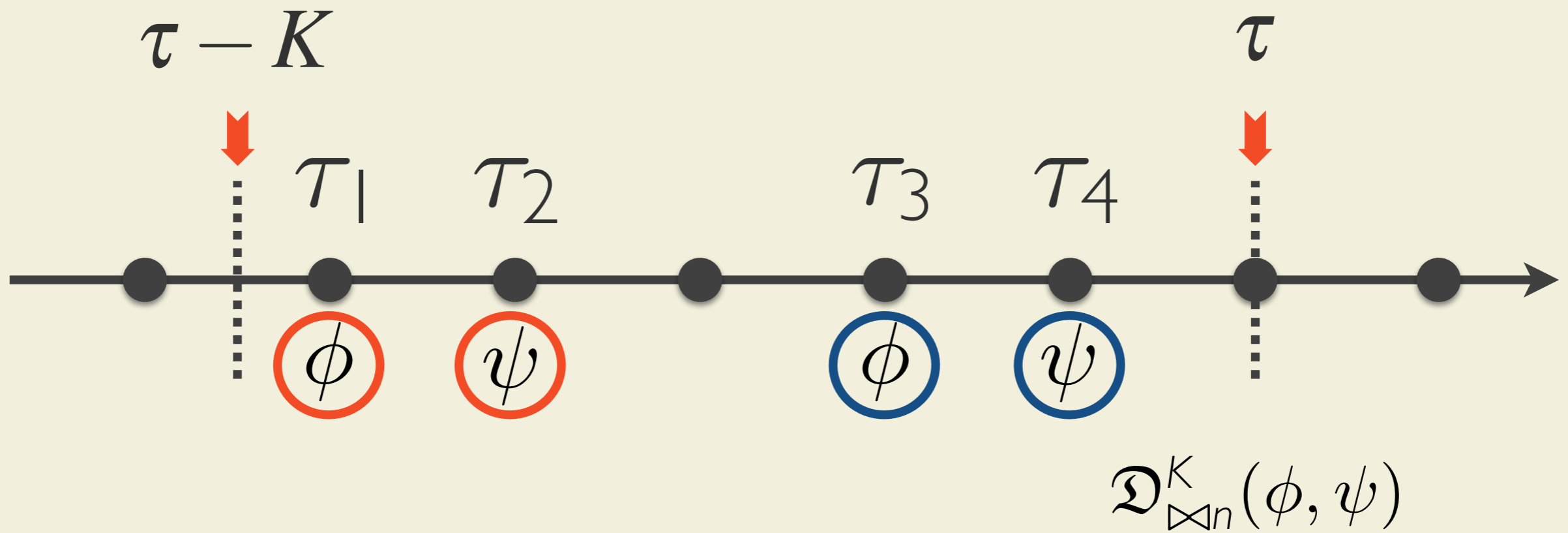
$$G(A \rightarrow \mathcal{D}_{<3}^{720}(C_{start}, C_{end}))$$



# SOLOIST



# SOLOIST

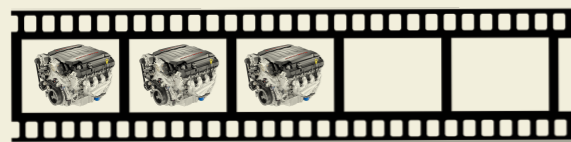


$$\frac{(\tau_2 - \tau_1) + (\tau_4 - \tau_3)}{2} \bowtie n$$

# Bounded Satisfiability Checking

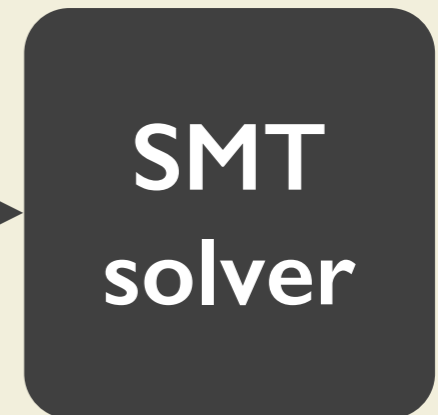
Execution trace

SOLOIST



$\wedge$

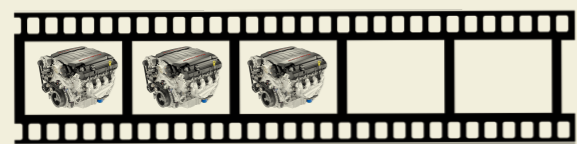
$\neg \varphi$



# Bounded Satisfiability Checking

Execution trace

SOLOIST



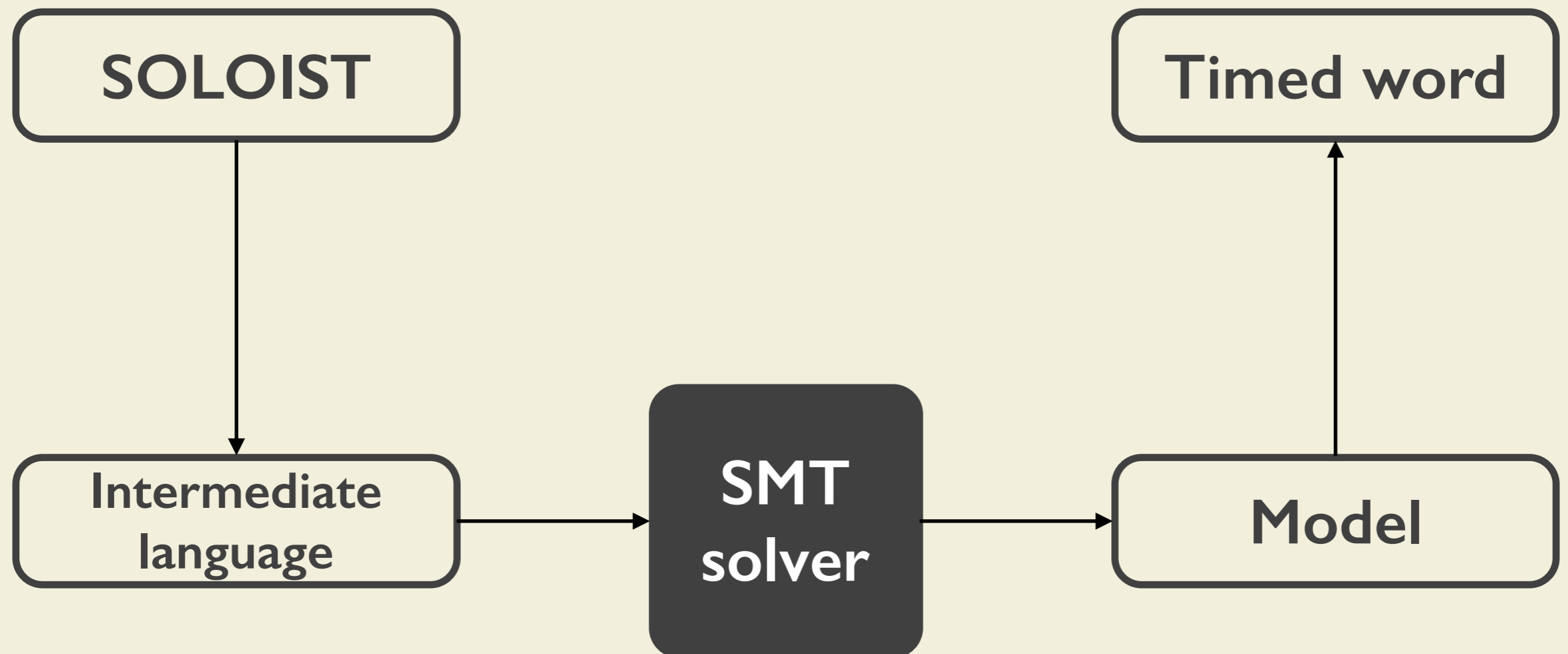
$\wedge$

$\neg \varphi$



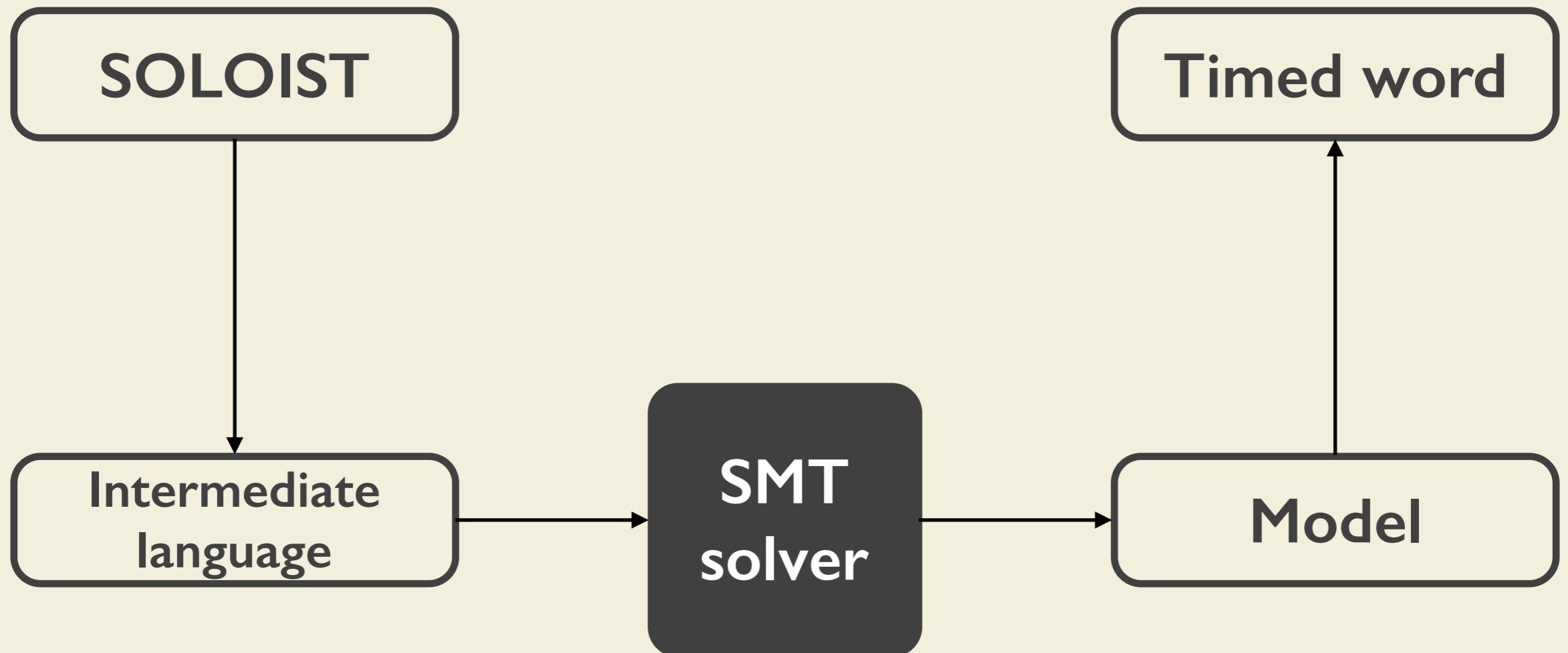
I don't speak SOLOIST!

# SOLOIST Satisfiability



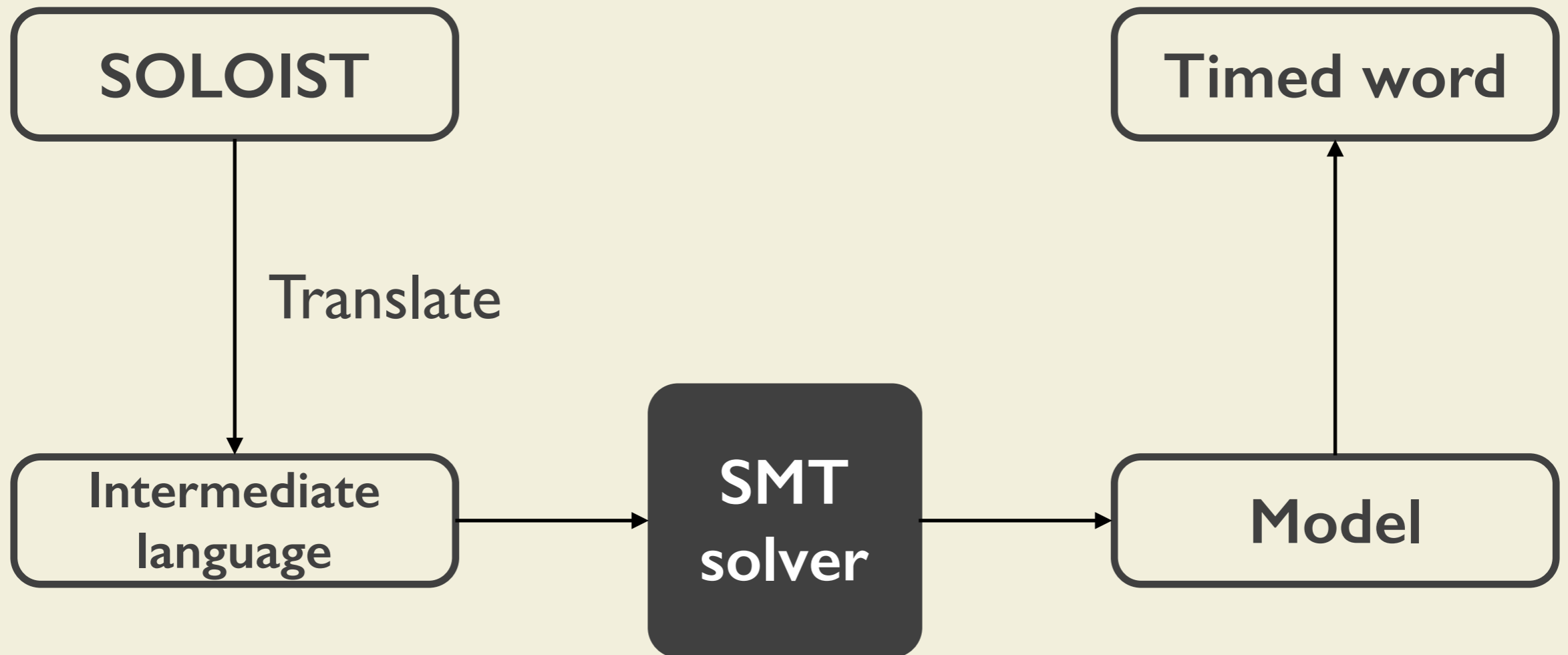
# SOLOIST Satisfiability

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



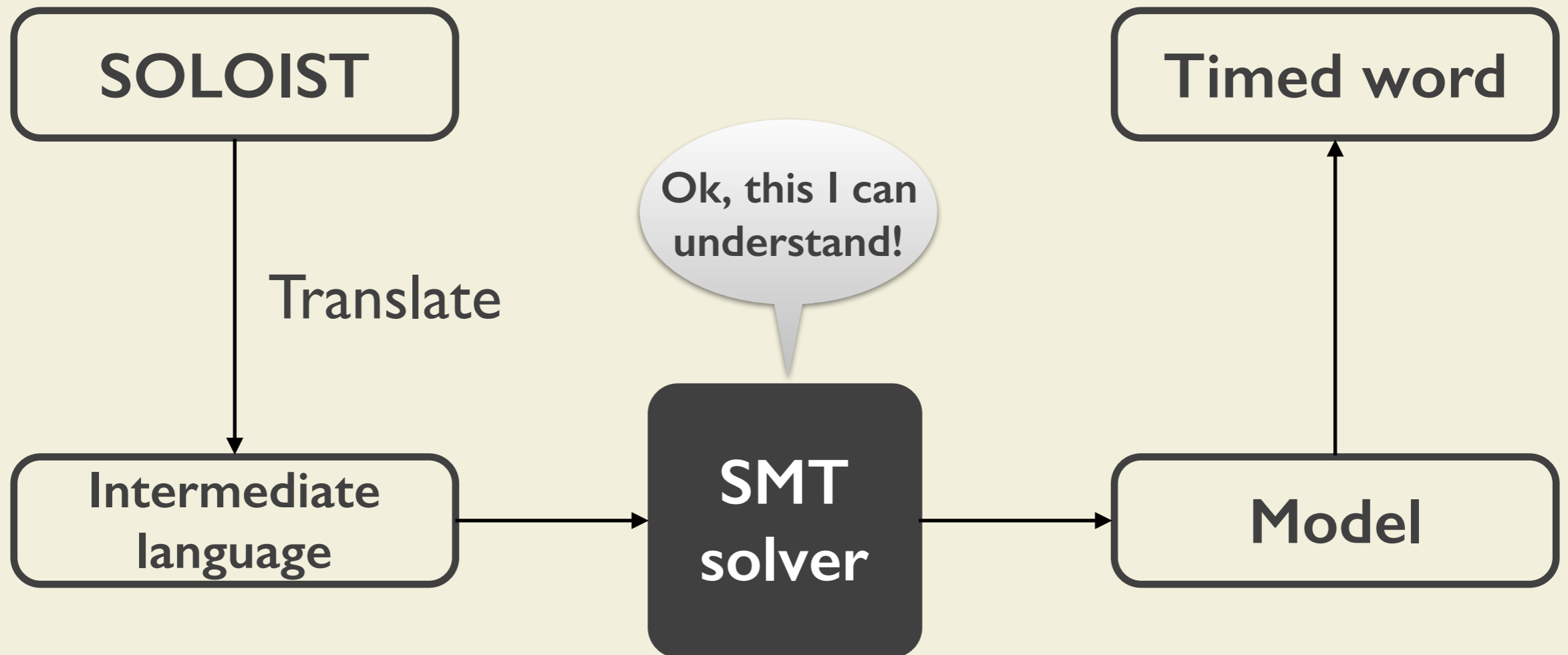
# SOLOIST Satisfiability

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



# SOLOIST Satisfiability

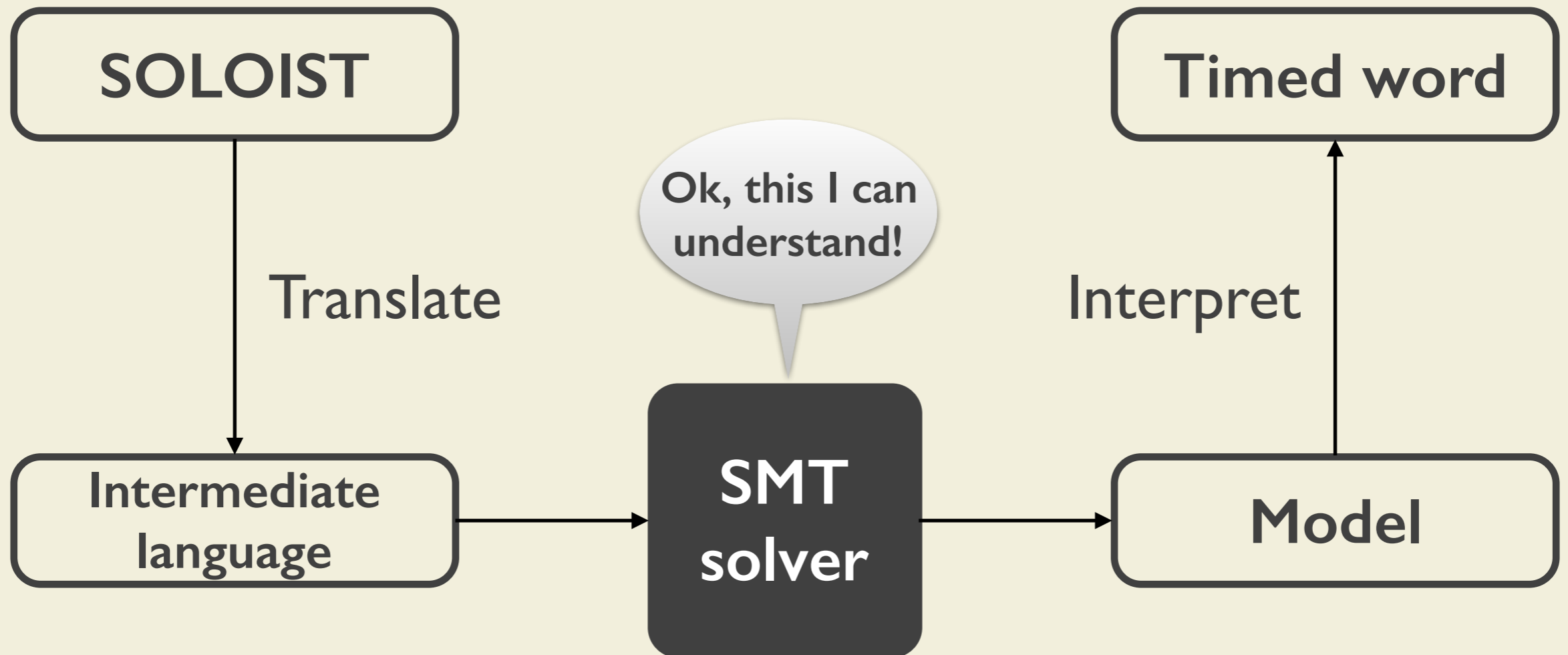
$$e_{<5}^{100}(p)$$



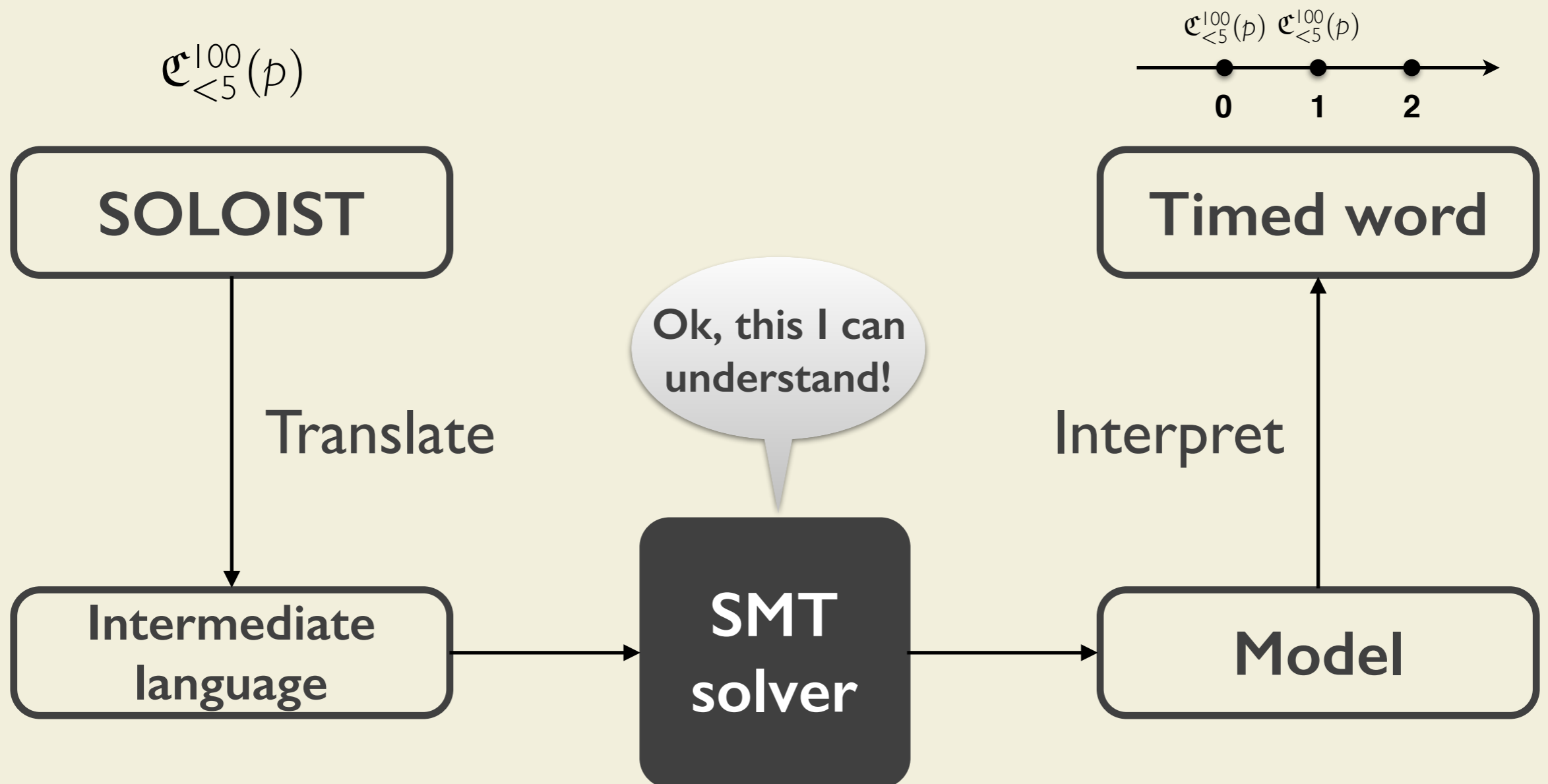


# SOLOIST Satisfiability

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



# SOLOIST Satisfiability



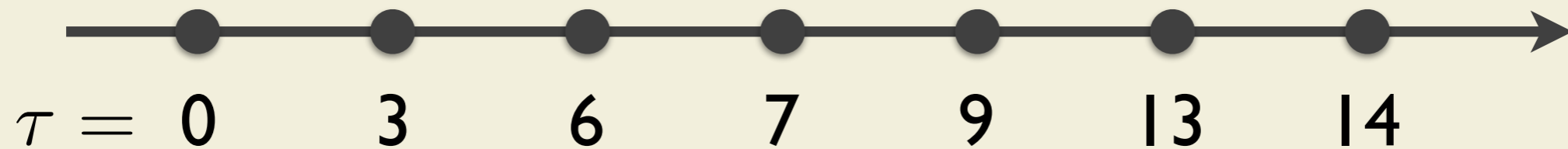
# Intermediate Language

# Intermediate Language

**CLTLB(D) - Constraint Linear Temporal Logic**

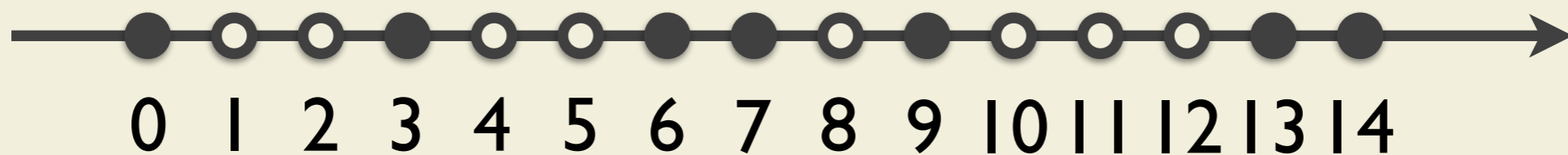
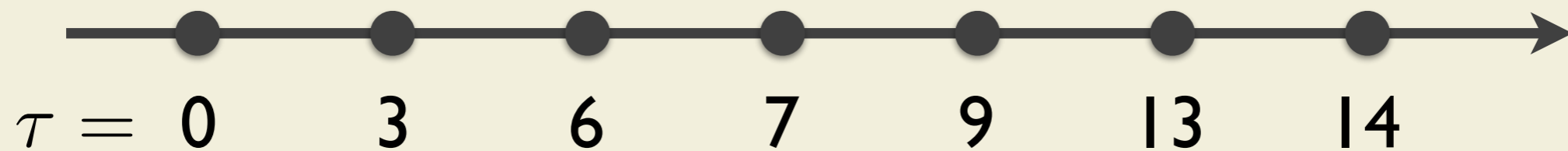
# Intermediate Language

CLTLB(D) - Constraint Linear Temporal Logic



# Intermediate Language

CLTLB(D) - Constraint Linear Temporal Logic



# Intermediate Language

# Intermediate Language

**QF-EUFIDL - Quantifier-free Integer difference logic  
with equality and uninterpreted functions**



# Intermediate Language

**QF-EUFIDL - Quantifier-free Integer difference logic  
with equality and uninterpreted functions**



# QF-EUFIDL

# QF-EUFIDL

$$\phi ::= p \mid t = t \mid t <_d t \mid \neg \phi \mid \phi \vee \phi$$

# QF-EUFIDL

$$\phi ::= p \mid t = t \mid t <_d t \mid \neg\phi \mid \phi \vee \phi$$

$$\{P_0, P_1, \dots\} \equiv \llbracket P \rrbracket : \mathbb{Z}_0^+ \rightarrow \{\top, \perp\}$$

# QF-EUFIDL

$$\phi ::= p \mid t = t \mid t <_d t \mid \neg\phi \mid \phi \vee \phi$$

$$\{P_0, P_1, \dots\} \equiv \llbracket P \rrbracket : \mathbb{Z}_0^+ \rightarrow \{\top, \perp\}$$

$$\{T_0, T_1, \dots\} \equiv \llbracket T \rrbracket : \mathbb{Z}_0^+ \rightarrow \mathbb{Z}$$

# QF-EUFIDL

$$\phi ::= p \mid t = t \mid t <_d t \mid \neg\phi \mid \phi \vee \phi$$

$$\{P_0, P_1, \dots\} \equiv \llbracket P \rrbracket : \mathbb{Z}_0^+ \rightarrow \{\top, \perp\} \quad \text{Predicate}$$

$$\{T_0, T_1, \dots\} \equiv \llbracket T \rrbracket : \mathbb{Z}_0^+ \rightarrow \mathbb{Z}$$

# QF-EUFIDL

$$\phi ::= p \mid t = t \mid t <_d t \mid \neg\phi \mid \phi \vee \phi$$

$$\{P_0, P_1, \dots\} \equiv \llbracket P \rrbracket : \mathbb{Z}_0^+ \rightarrow \{\top, \perp\} \quad \text{Predicate}$$

$$\{T_0, T_1, \dots\} \equiv \llbracket T \rrbracket : \mathbb{Z}_0^+ \rightarrow \mathbb{Z} \quad \text{Arithmetical variable}$$

# Translation

 $\mathcal{T}$ 

**timing information**

 $\neg p$ 

**propositional operators**

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

**metric temporal operators**

 $\mathbf{e}_{<2}^K \phi$ 

**aggregate operators**



# Translation

 $\mathcal{T}$ 

**timing information**

 $\neg\phi$ 

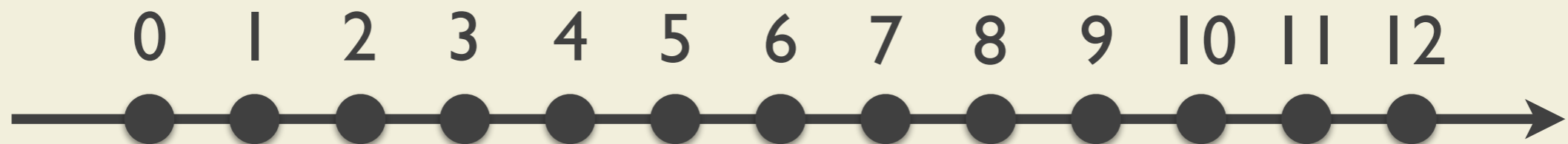
**propositional operators**

 $\mathbf{c}_{<2}^k\phi$ 

**counting modality**

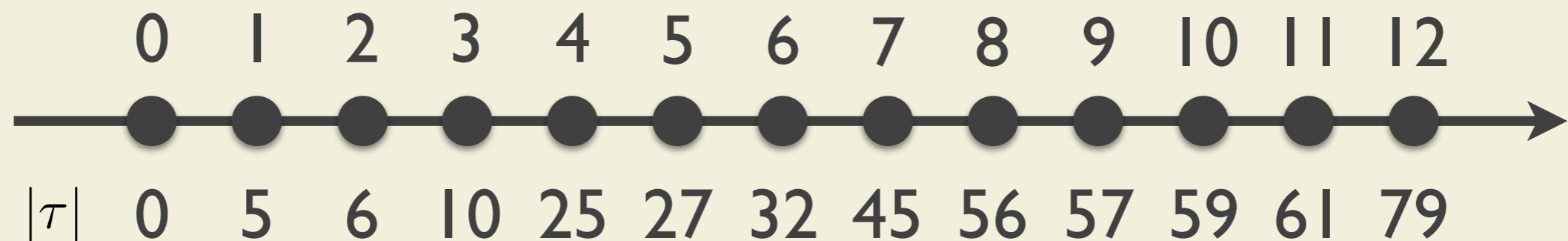
# Timing Information

$$|\tau|_i < |\tau|_{i+1}, \forall i \in \{0 \dots H - 1\}$$



# Timing Information

$$|\tau|_i < |\tau|_{i+1}, \forall i \in \{0 \dots H - 1\}$$

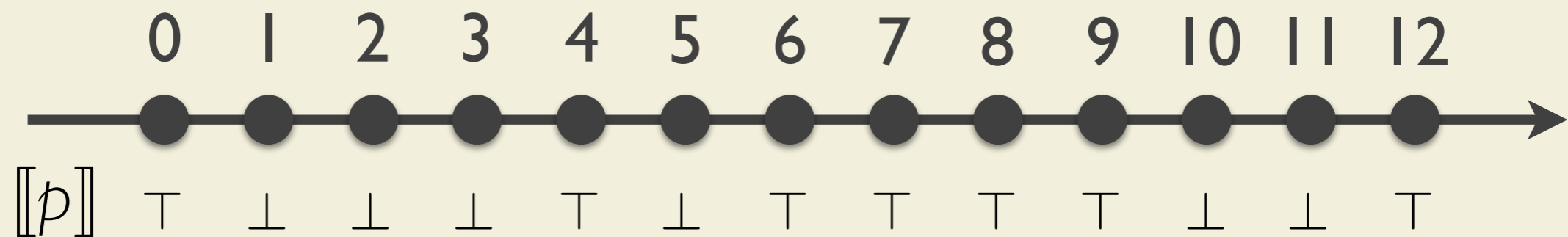


# Boolean Operators



# Boolean Operators

$$\llbracket p \rrbracket_i \leftrightarrow p(i), \forall i \in \{0 \dots H\}$$



# Boolean Operators

$$\llbracket p \rrbracket_i \leftrightarrow p(i), \forall i \in \{0 \dots H\}$$

$$\llbracket \neg p \rrbracket_i \leftrightarrow \neg p(i), \forall i \in \{0 \dots H\}$$

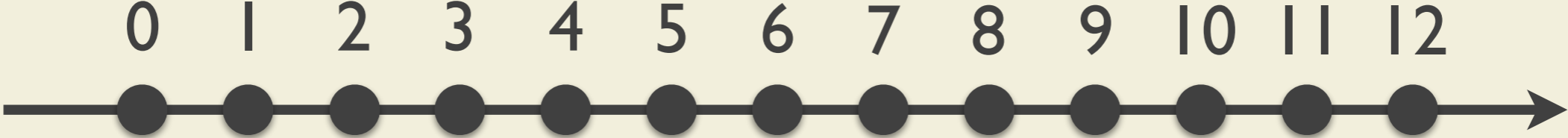
	0	1	2	3	4	5	6	7	8	9	10	11	12
$\llbracket p \rrbracket$	T	⊥	⊥	⊥	T	⊥	T	T	T	T	⊥	⊥	T
$\llbracket \neg p \rrbracket$	⊥	T	T	T	⊥	T	⊥	⊥	⊥	⊥	T	T	⊥

# Boolean Operators

$$\llbracket p \rrbracket_i \leftrightarrow p(i), \forall i \in \{0 \dots H\}$$

$$\llbracket \neg p \rrbracket_i \leftrightarrow \neg p(i), \forall i \in \{0 \dots H\}$$

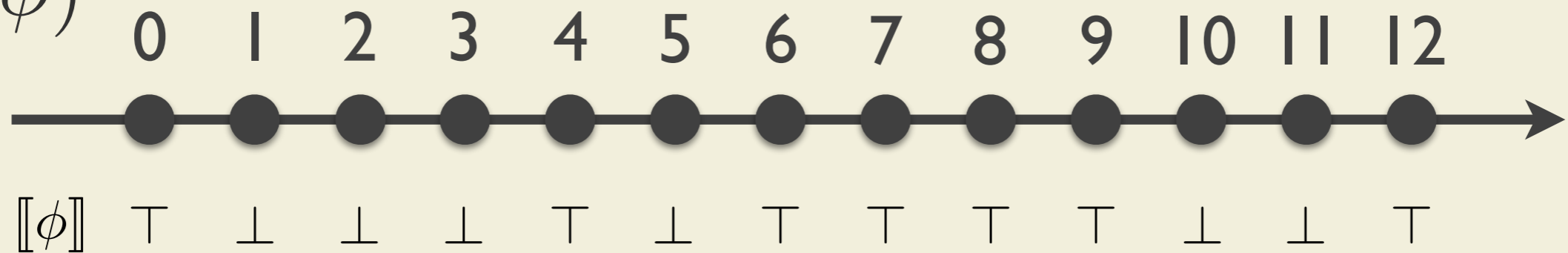
$$\llbracket \phi \wedge \psi \rrbracket_i \leftrightarrow \llbracket \phi \rrbracket_i \wedge \llbracket \psi \rrbracket_i, \forall i \in \{0 \dots H\}$$



	0	1	2	3	4	5	6	7	8	9	10	11	12
$\llbracket p \rrbracket$	T	⊥	⊥	⊥	T	⊥	T	T	T	T	⊥	⊥	T
$\llbracket \neg p \rrbracket$	⊥	T	T	T	⊥	T	⊥	⊥	⊥	⊥	T	T	⊥
$\llbracket p \wedge \neg p \rrbracket$	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥	⊥

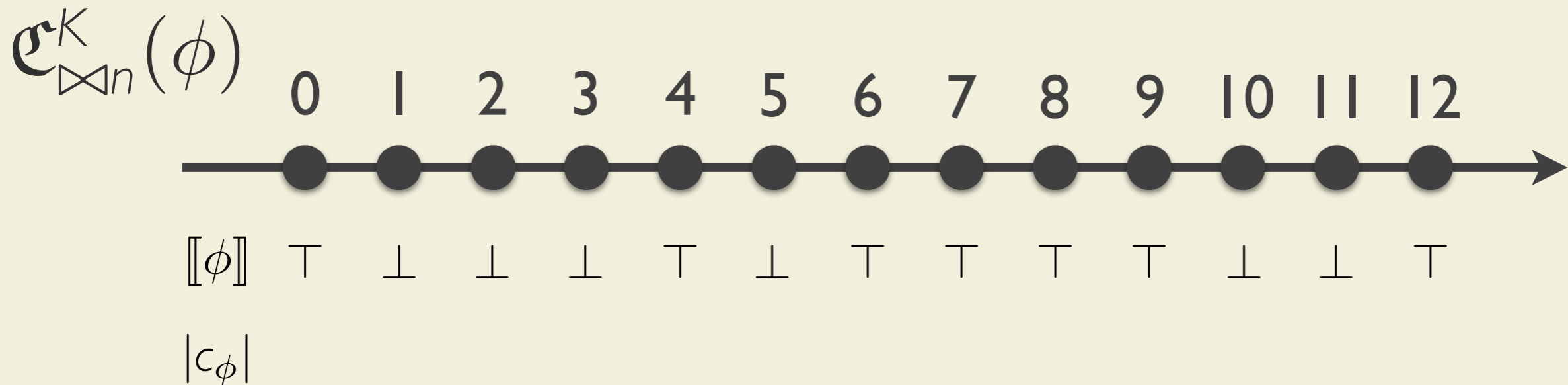
# Counting Modality

$\mathcal{C}_n^K(\phi)$





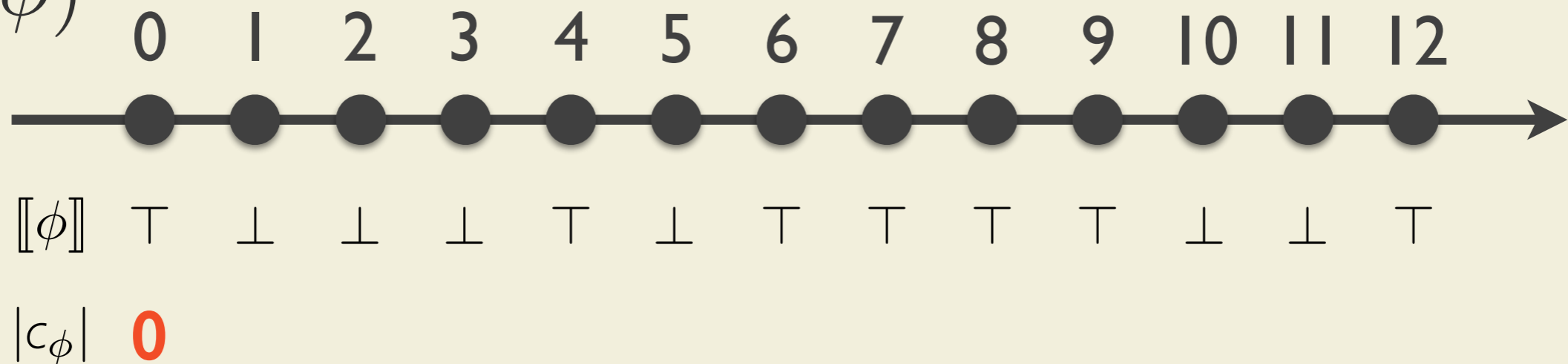
# Counting Modality



# Counting Modality

$$|c_\phi|_0 = 0$$

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



# Counting Modality

$$|c_\phi|_0 = 0$$

$$[[\phi]]_i \rightarrow (|c_\phi|_{i+1} = (|c_\phi|_i + 1)), \forall i \in \{0 \dots H - 1\}$$

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



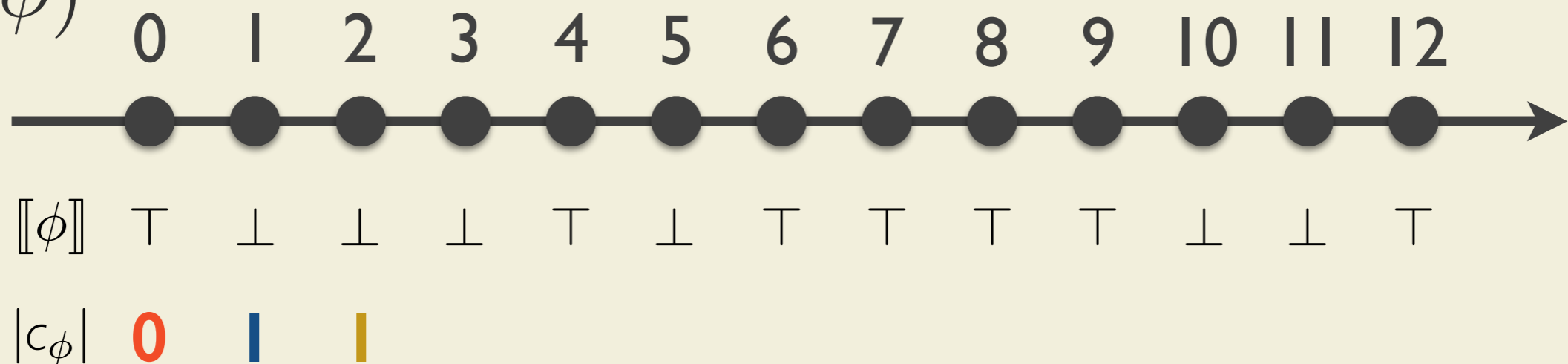
# Counting Modality

$$|c_\phi|_0 = 0$$

$$[[\phi]]_i \rightarrow (|c_\phi|_{i+1} = (|c_\phi|_i + 1)), \forall i \in \{0 \dots H - 1\}$$

$$\neg [[\phi]]_i \rightarrow (|c_\phi|_{i+1} = |c_\phi|_i), \forall i \in \{0 \dots H - 1\}$$

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



# Counting Modality

$$|c_\phi|_0 = 0$$

$$[[\phi]]_i \rightarrow (|c_\phi|_{i+1} = (|c_\phi|_i + 1)), \forall i \in \{0 \dots H - 1\}$$

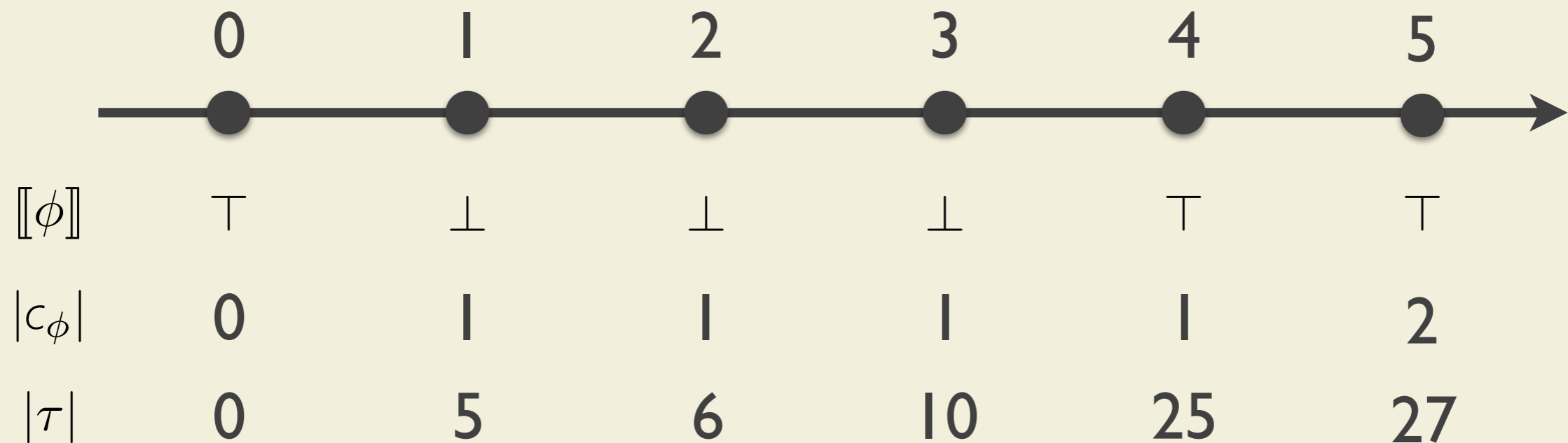
$$\neg [[\phi]]_i \rightarrow (|c_\phi|_{i+1} = |c_\phi|_i), \forall i \in \{0 \dots H - 1\}$$

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

	0	1	2	3	4	5	6	7	8	9	10	11	12
$[[\phi]]$	T	⊥	⊥	⊥	T	⊥	T	T	T	T	⊥	⊥	T
$ c_\phi $	0	1	1	1	1	2	2	3	4	5	6	6	6

# Counting Modality

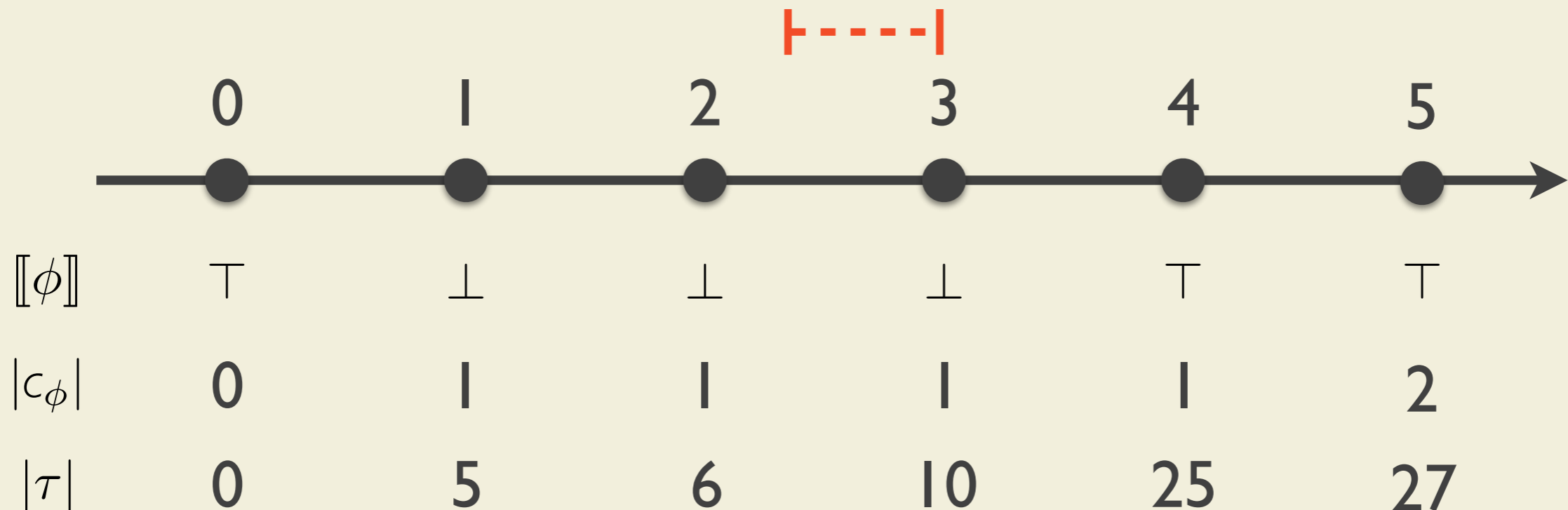
$$\llbracket \mathcal{C}_{<3}^9(\phi) \rrbracket_3 \leftrightarrow$$



# Counting Modality

$$|c_\phi|_4 - |c_\phi|_3 < 3 \wedge |\tau|_3 - |\tau|_2 > 9 \wedge |\tau|_3 - |\tau|_3 \leq 9$$

$$[\mathcal{E}_{<3}^9(\phi)]_3 \leftrightarrow$$

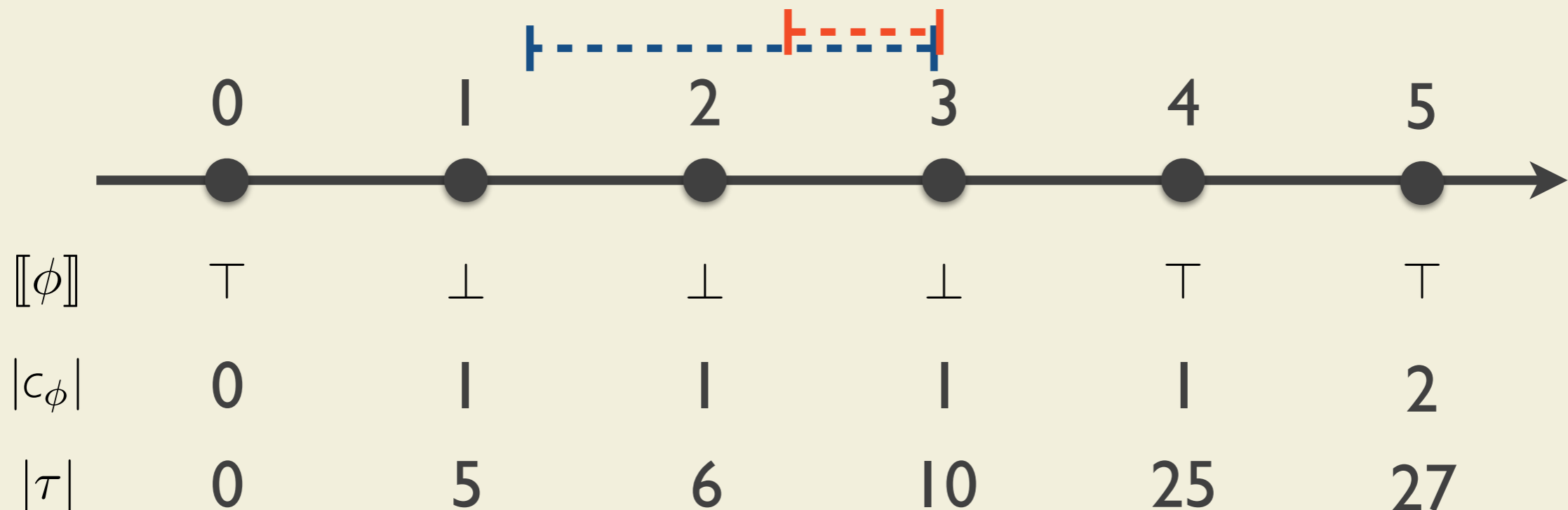


# Counting Modality

$$|c_\phi|_4 - |c_\phi|_3 < 3 \wedge |\tau|_3 - |\tau|_2 > 9 \wedge |\tau|_3 - |\tau|_3 \leq 9$$

∨

$$\llbracket \mathcal{C}_{<3}^9(\phi) \rrbracket_3 \leftrightarrow |c_\phi|_4 - |c_\phi|_2 < 3 \wedge |\tau|_3 - |\tau|_1 > 9 \wedge |\tau|_3 - |\tau|_2 \leq 9$$





# Counting Modality

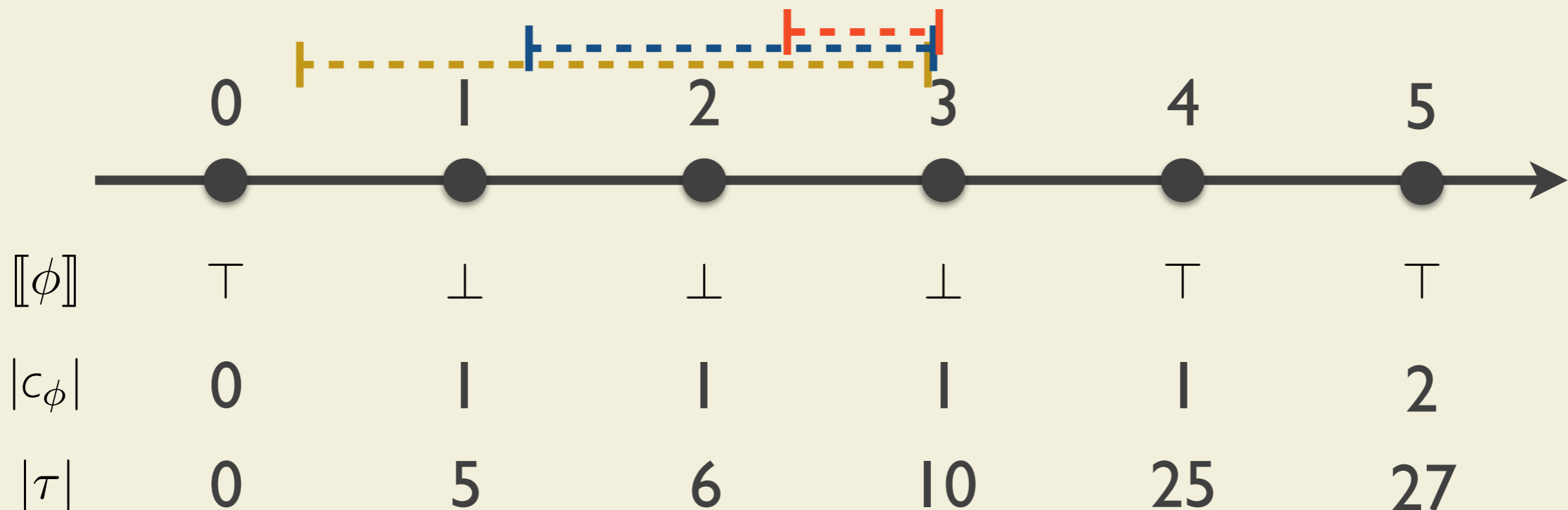
$$|c_\phi|_4 - |c_\phi|_3 < 3 \wedge |\tau|_3 - |\tau|_2 > 9 \wedge |\tau|_3 - |\tau|_3 \leq 9$$

∨

$$[\mathcal{E}_{<3}^9(\phi)]_3 \leftrightarrow |c_\phi|_4 - |c_\phi|_2 < 3 \wedge |\tau|_3 - |\tau|_1 > 9 \wedge |\tau|_3 - |\tau|_2 \leq 9$$

∨

$$|c_\phi|_4 - |c_\phi|_1 < 3 \wedge |\tau|_3 - |\tau|_0 > 9 \wedge |\tau|_3 - |\tau|_1 \leq 9$$



# Counting Modality

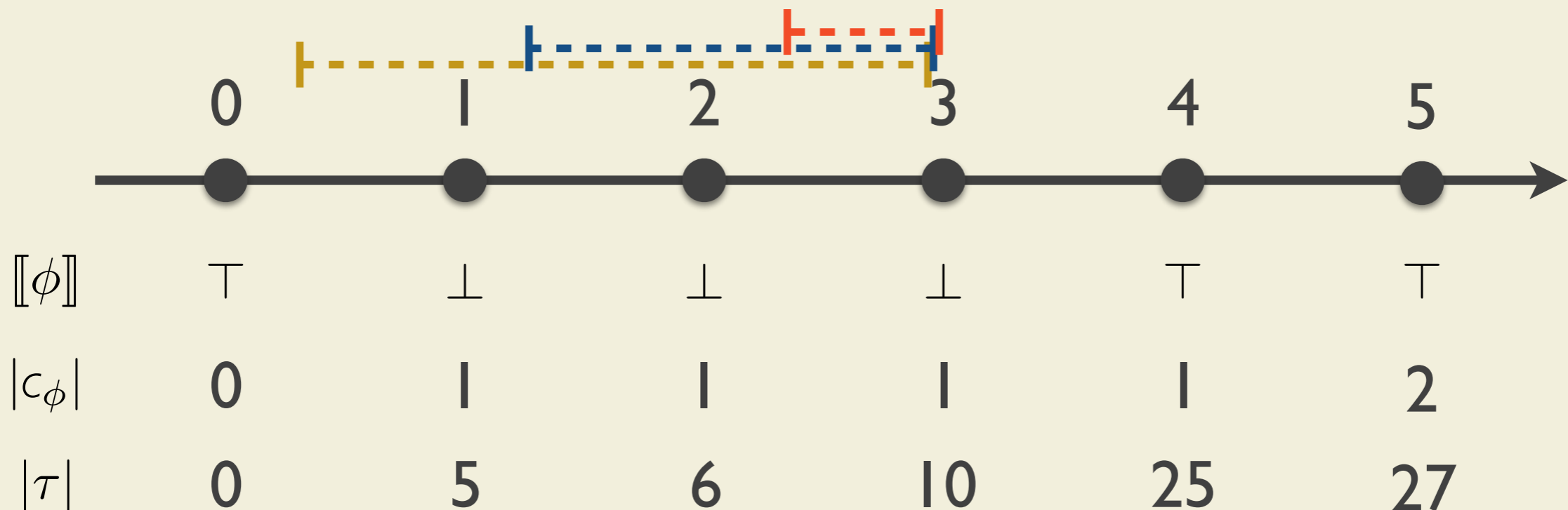
$$|c_\phi|_4 - |c_\phi|_3 < 3 \wedge |\tau|_3 - |\tau|_2 > 9 \wedge |\tau|_3 - |\tau|_3 \leq 9$$

∨

$$[\mathcal{E}_{<3}^9(\phi)]_3 \leftrightarrow |c_\phi|_4 - |c_\phi|_2 < 3 \wedge |\tau|_3 - |\tau|_1 > 9 \wedge |\tau|_3 - |\tau|_2 \leq 9$$

∨

$$|c_\phi|_4 - |c_\phi|_1 < 3 \wedge |\tau|_3 - |\tau|_0 > 9 \wedge |\tau|_3 - |\tau|_1 \leq 9$$



# Counting Modality

$$\begin{aligned}
 \llbracket \mathbf{c}_{\bowtie n}^K(\phi) \rrbracket_i &\leftrightarrow \bigvee_{z=0}^{\min\{i,K\}} (|c_\phi|_{i+1} - |c_\phi|_{i-z} \bowtie n \wedge \\
 &\quad |T|_i - |T|_{i-z-1} > K \wedge \\
 &\quad |T|_i - |T|_{i-z} \leq K)
 \end{aligned}$$

# Complexity

# Complexity

$\lambda$  size of the formula

$\mu$  largest constant

$H$  number of relevant events

# Complexity

$\lambda$  size of the formula

$\mu$  largest constant

$H$  number of relevant events

**Translation size:**  $\mathcal{O}(\lambda \log(\mu) H^3)$

# Complexity

$\lambda$  size of the formula

$\mu$  largest constant

$H$  number of relevant events

**Translation size:**  $\mathcal{O}(\lambda \log(\mu) H^3)$

**QF-EUFIDL satisfiability: NP-complete**

# Complexity

$\lambda$  size of the formula

$\mu$  largest constant

$H$  number of relevant events

**Translation size:**  $\mathcal{O}(\lambda \log(\mu) H^3)$

**QF-EUFIDL satisfiability: NP-complete**

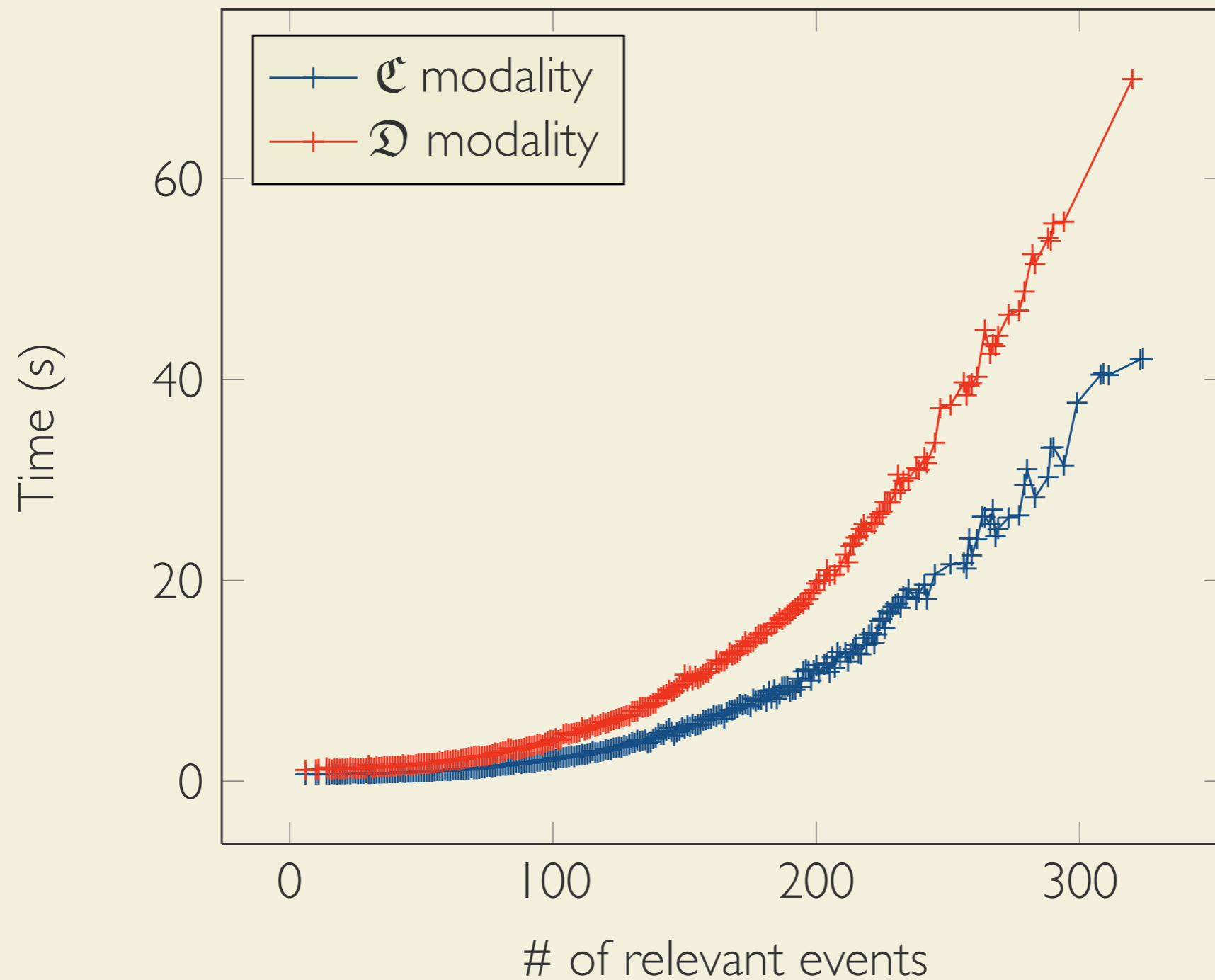
**SOLOIST trace checking is NP-complete**



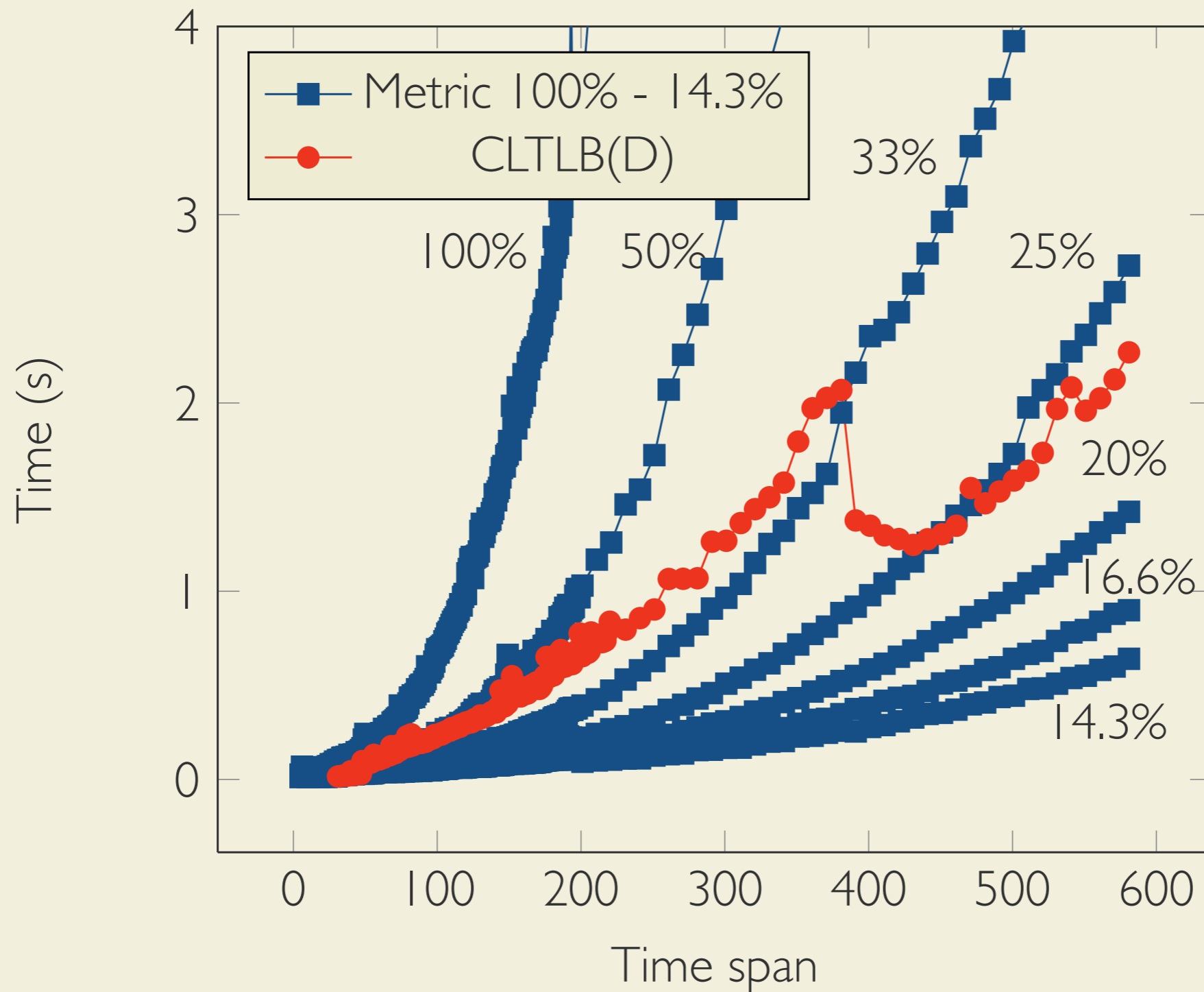
# Implementation and Evaluation

- Implemented as a Common Lisp plugin for the ZOT verification toolkit
- 30.000 traces were synthesized using the Process Log Generator tool

# Scalability - Time Performance



# Sparseness - Time Performance



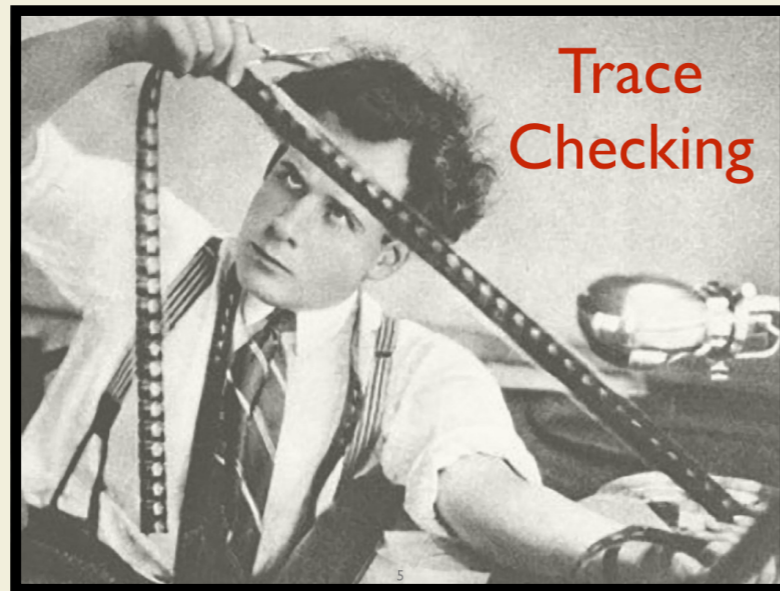
# Application to a Real Dataset

- We checked service composition from the “S-Cube Use Case Repository”
- 9796 execution traces

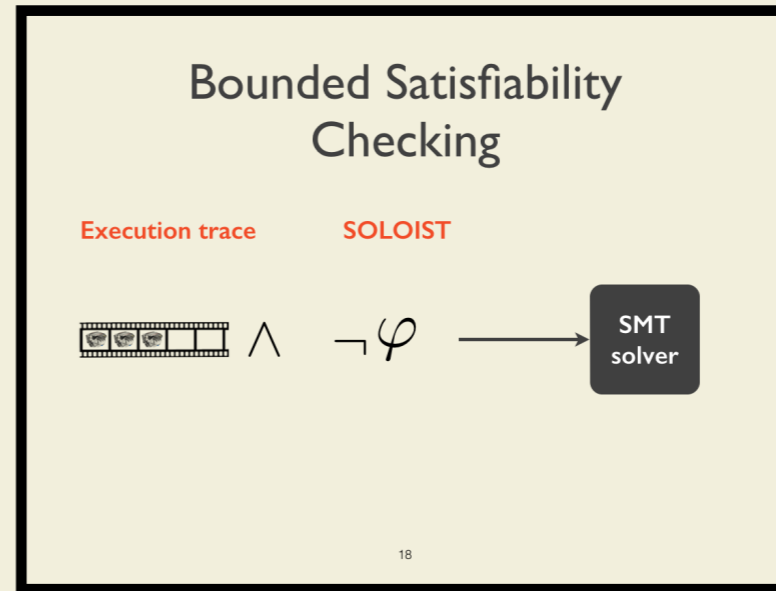
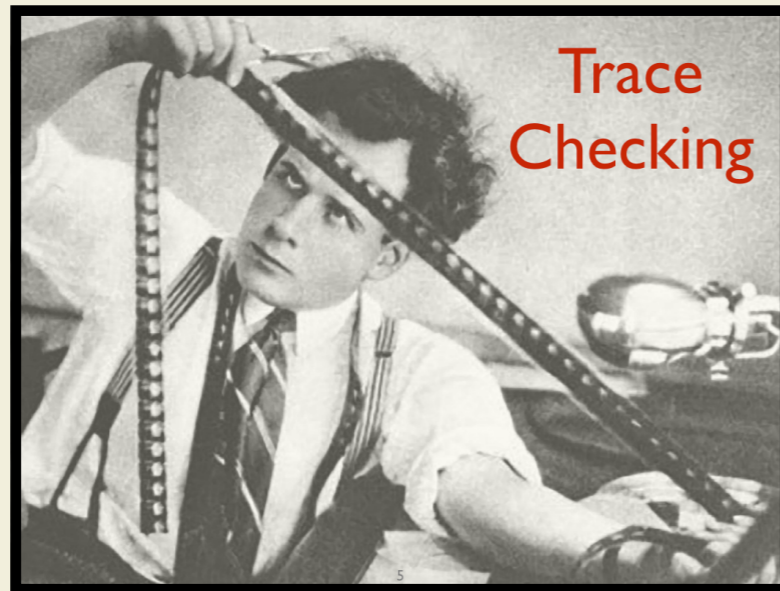
	<b>Average time (s)</b>	<b>Average memory (MB)</b>
<b>C modality</b>	0.672±0.035	125.7±0.476
<b>D modality</b>	0.813±0.032	127.7±0.476

# SMT-based Checking of SOLOIST over Sparse Traces

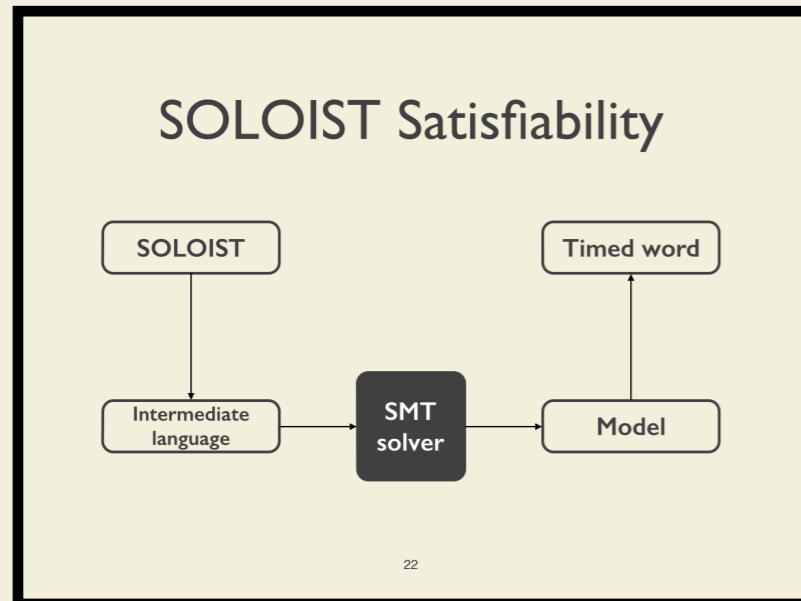
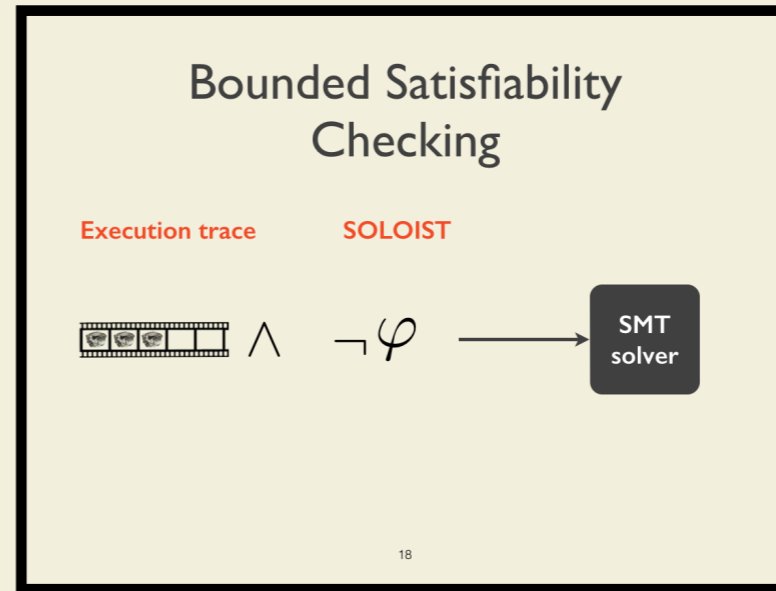
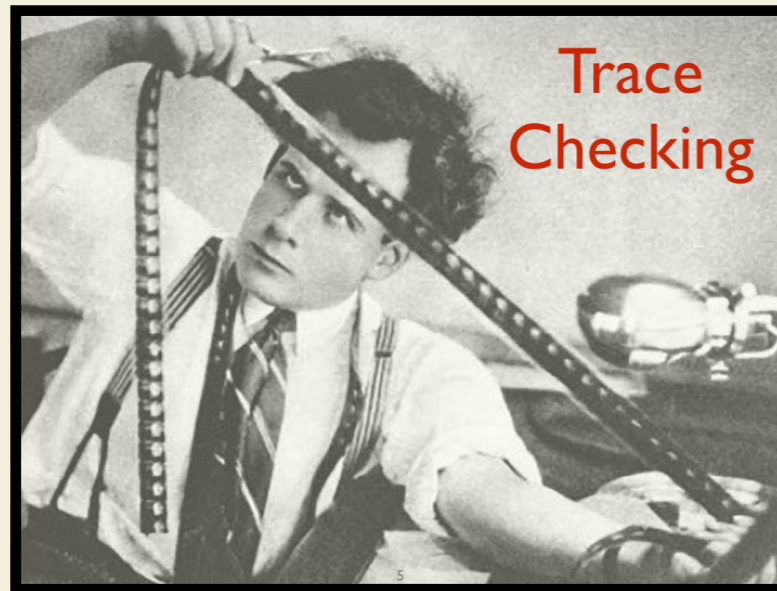
# SMT-based Checking of SOLOIST over Sparse Traces



# SMT-based Checking of SOLOIST over Sparse Traces

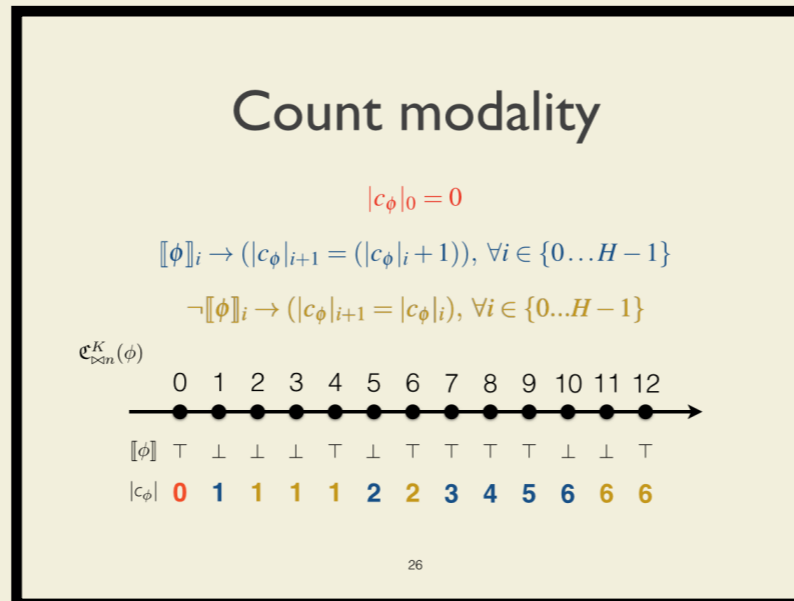
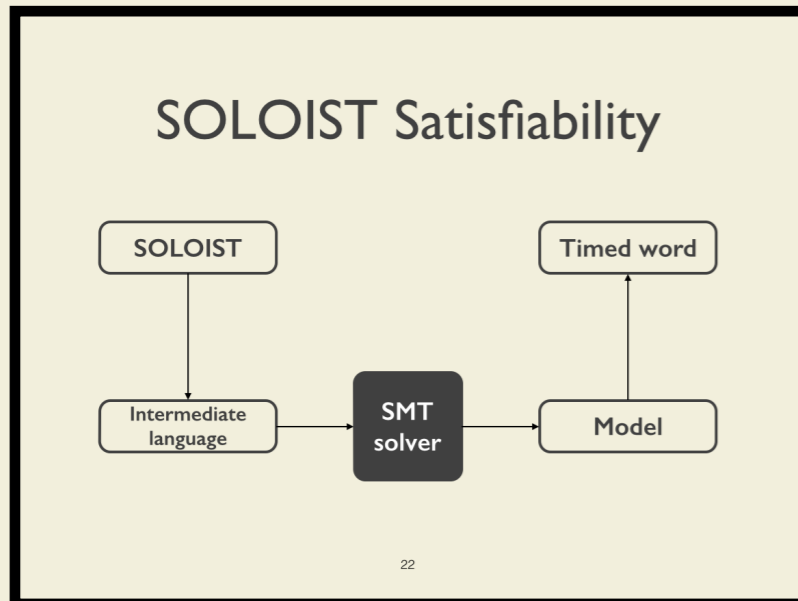
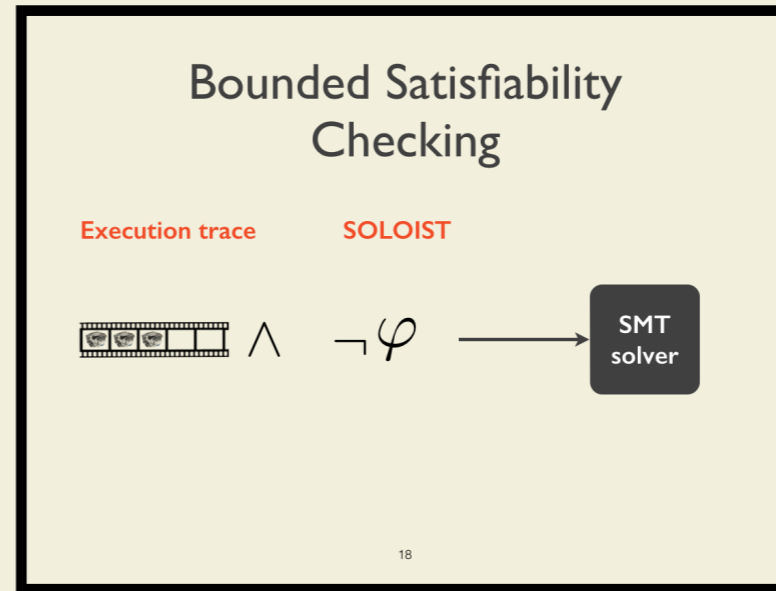
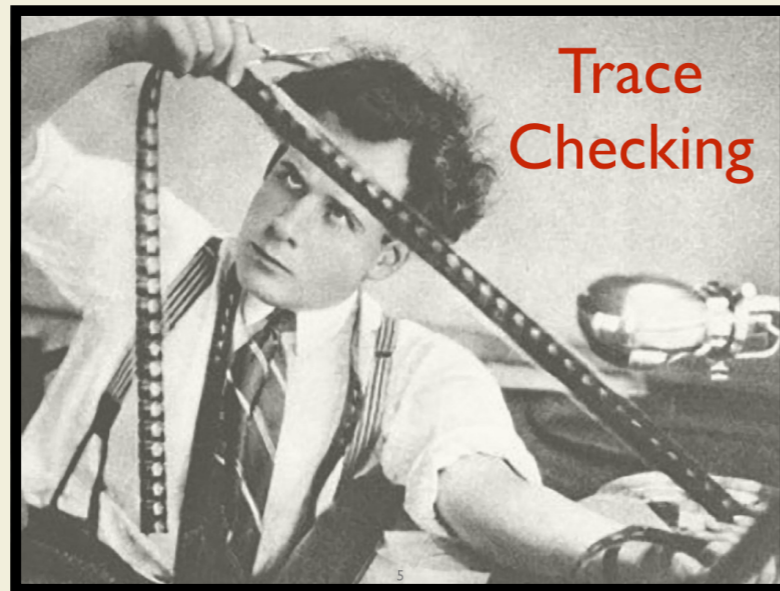


# SMT-based Checking of SOLOIST over Sparse Traces

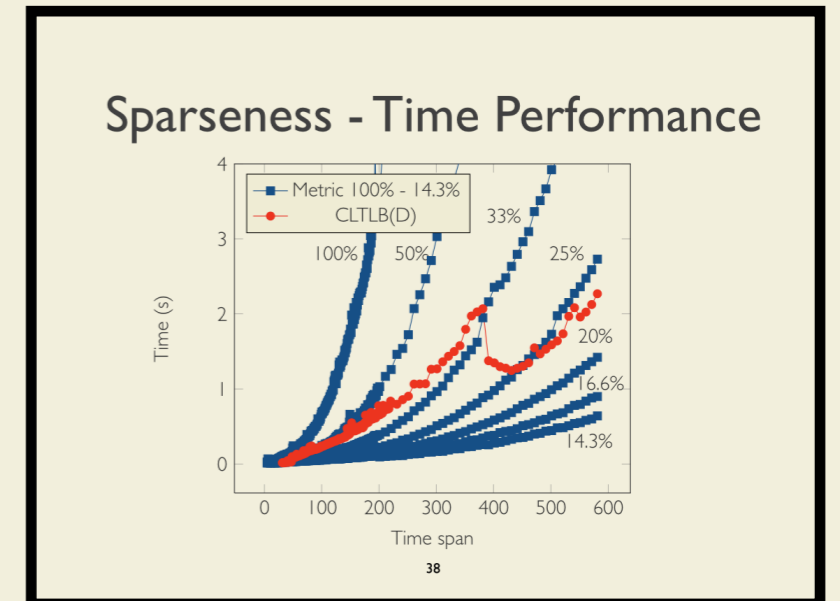
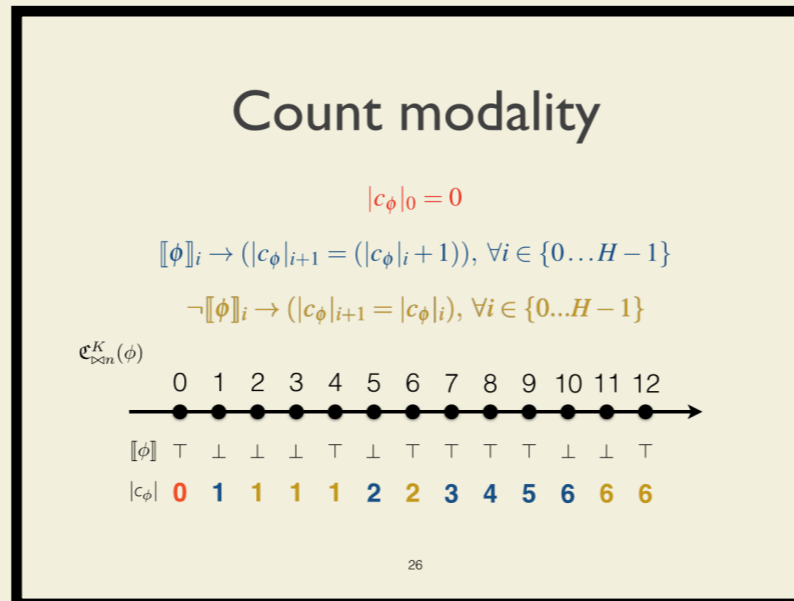
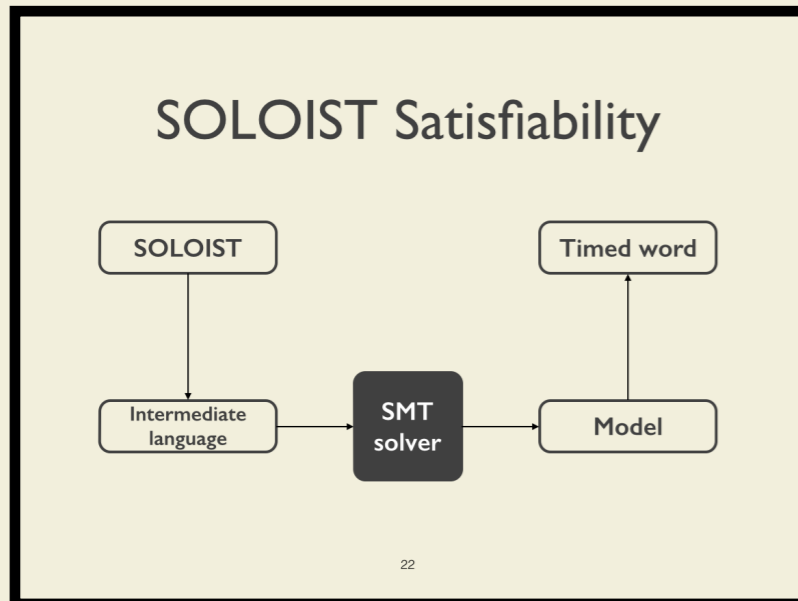
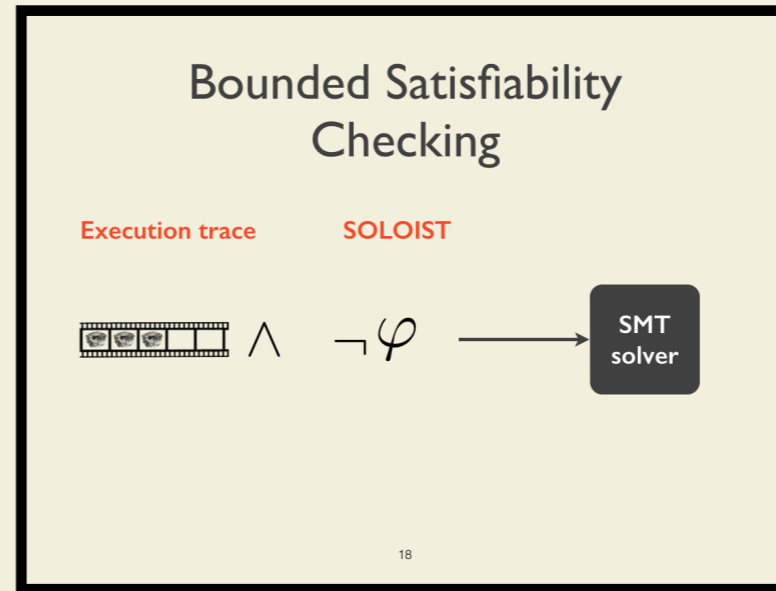
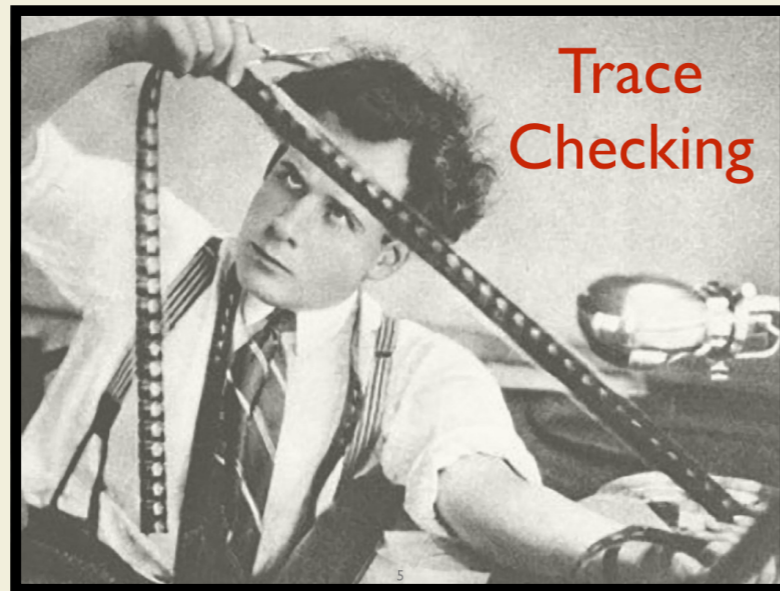




# SMT-based Checking of SOLOIST over Sparse Traces



# SMT-based Checking of SOLOIST over Sparse Traces



# Current and Future Work

- Formalize new properties: **Towards the Formalization of Properties of Cloud-based Elastic Systems.**  
To appear at PESOS 2014 (ICSE 2014)
- Extend the approach for very large logs using MapReduce
- Application to run-time verification

# SMT-based Checking of SOLOIST over Sparse Traces

Srđan Krstić

with

Marcello Bersani, Domenico Bianculli, Carlo Ghezzi and Pierluigi San Pietro

