上海交通大学
SHANGHAI JIAO TONG UNIVERSITY

SJTU

基于MagicSets的并行时序推理方法与实现
中期检查汇报

报告人：赵楷越 韦东良

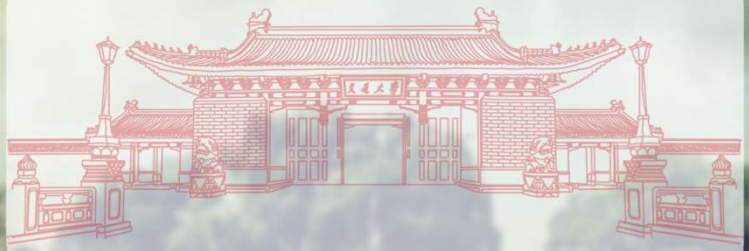
2024年7月21日

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

目录

01

项目进展与成果



进展：认识DatalogMTL

流数据广泛存在于金融交易、网络异常检测、设备预警、智能制造等领域。**基于流数据的符号推理**为上述领域的众多应用提供了有力支撑。

处理流数据

- **金融交易**
- **网络信息**
- **制造设备**
- **…**
- **(更多时序信息)**

A Simple DatalogMTL Example:

$$Authorised(x) \leftarrow \diamondminus_{[0,2]} NegativeLFT(x) \wedge \boxminus_{[0,15]} FullyVaccinated(x).$$

DatalogMTL是一种新兴的支持流数据推理的逻辑编程语言，相较于命令式编程，使用DatalogMTL规则语言能够更好地**固化专家知识**，**降低沟通和开发成本**。

进展：认识DatalogMTL语言与推理器MeTeoR

MeTeoR: Practical Reasoning in Datalog with Metric Temporal Operators

Dingmin Wang[1], Pan Hu[1,2], Przemysław Andrzej Wałęga[1], Bernardo Cuenca Grau[1]

[1]Department of Computer Science, University of Oxford, UK
{dingmin.wang, przemyslaw.walega, bernardo.cuenca.grau}@cs.ox.ac.uk
[2]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China
pan.hu@sjtu.edu.cn

Abstract

DatalogMTL is an extension of Datalog with operators from metric temporal logic which has received significant attention in recent years. It is a highly expressive knowledge representation language that is well-suited for applications in temporal ontology-based query answering and stream processing. Reasoning in DatalogMTL is, however, of high computational complexity, making implementation challenging and hindering its adoption in practice. In this paper, we propose a novel approach for practical reasoning in DatalogMTL which combines materialisation (a.k.a. forward chaining) with automata-based techniques. We have implemented this approach in a reasoner called MeTeoR and evaluated its performance using a temporal extension of the Lehigh University Benchmark, and a benchmark based on real-world meteorological data. Our experiments show that MeTeoR is a scalable system which enables reasoning over complex temporal rules and datasets involving tens of millions of temporal facts.

Introduction

Temporal data is ubiquitous in many application scenarios, such as stock trading (Nait et al. 2013), network flow anomaly detection (Moser and Carle 2007), and equipment malfunction monitoring (Doherty, Kvarnström, and Heintz 2009). In order to represent knowledge and subsequently reason in the presence of such temporal data, Brandt et al. (2018) proposed DatalogMTL—an extension of Datalog (Ceri, Gottlob, and Tanca 1989) with operators from metric temporal logic (Koymans 1990) interpreted over the rational timeline. DatalogMTL is a powerful KR language, which has found applications in ontology-based query answering (Brandt et al. 2018; Kikot et al. 2018; Kalayci et al. 2018; Kougioumtzoglou 2019) and stream reasoning (Wałęga, Kaminski, and Cuenca Grau 2019). For example, the following DatalogMTL rule can be used to analyse equipment data:

$$\boxplus_{[0,1]} (\mathit{ExHeat}(x) \leftarrow \boxminus_{[0,1]} \mathit{Temp24}(x) \wedge \diamondminus_{[0,1]} \mathit{Temp41}(x).$$

The rule states that a device x has been under excessive heat continuously within a past interval of length 1 ($\boxminus_{[0,1]}$) if the temperature recorded in this interval was always above 24 degrees (*Temp24*) and also, at some point in this interval ($\diamondminus_{[0,1]}$), the temperature was above 41 degrees (*Temp41*).

Reasoning in DatalogMTL is, however, of high complexity, namely ExpSpace-complete (Brandt et al. 2018) and PSpace-complete with respect to data size (Wałęga et al. 2019), which makes reasoning in practice challenging. Thus, research has recently focused on establishing a suitable trade-off between expressive power and complexity of reasoning, by identifying lower-complexity fragments of DatalogMTL (Wałęga et al. 2020b; Wałęga, Zawidzki, and Cuenca Grau 2021) as well as studying alternative semantics with more favourable computational behaviour (Wałęga et al. 2020a; Ryzhikov, Wałęga, and Zakharyaschev 2019).

The design and implementation of practical reasoning algorithms for the full DatalogMTL language, however, remains a largely unexplored area—something that has prevented its widespread adoption in applications. In particular, the only implementation we are aware of is the prototype by Brandt et al. (2018), which is limited to non-recursive DatalogMTL programs. This is in stark contrast with plain Datalog, for which a plethora of systems have been developed and successfully deployed in practice (Motik et al. 2014; Carral et al. 2019; Bellomarini, Sallinger, and Gottlob 2018).

In this paper, we present the first practical reasoning algorithm for the full DatalogMTL language, which combines materialisation (a.k.a. forward chaining) and automata-based reasoning. On the one hand, materialisation is the reasoning paradigm adopted in most Datalog systems (Ceri et al. 2007); to check fact entailment in this setting, one first computes in a forward chaining manner all facts logically entailed by the input program and dataset, and then verifies whether the input fact is included amongst the entailed facts. A direct implementation of materialisation-based reasoning in DatalogMTL is, however, problematic since forward chaining may require infinitely many rounds of rule applications (Wałęga, Zawidzki, and Cuenca Grau 2021). On the other hand, Wałęga et al. (2019) introduced a decision procedure for DatalogMTL which relies on constructing Büchi automata and checking non-emptiness of their languages. This procedure has been introduced for obtaining tight complexity bounds, and not with efficient implementation in mind; in particular, the constructed automata are of exponential size, which makes direct implementations impractical.

对论文的研读

$$
\begin{aligned}
&\mathfrak{I}, t \models \diamondminus_{\varrho} M && \text{iff } \mathfrak{I}, t' \models M \text{ for some } t' \text{ with } t - t' \in \varrho \\
&\mathfrak{I}, t \models \diamondplus_{\varrho} M && \text{iff } \mathfrak{I}, t' \models M \text{ for some } t' \text{ with } t' - t \in \varrho \\
&\mathfrak{I}, t \models \boxminus_{\varrho} M && \text{iff } \mathfrak{I}, t' \models M \text{ for all } t' \text{ with } t - t' \in \varrho \\
&\mathfrak{I}, t \models \boxplus_{\varrho} M && \text{iff } \mathfrak{I}, t' \models M \text{ for all } t' \text{ with } t' - t \in \varrho \\
&\mathfrak{I}, t \models M_1 \mathcal{S}_{\varrho} M_2 && \text{iff } \mathfrak{I}, t' \models M_2 \text{ for some } t' \text{ with } t - t' \in \varrho \\
&&& \quad \text{and } \mathfrak{I}, t'' \models M_1 \text{ for all } t'' \in (t', t) \\
&\mathfrak{I}, t \models M_1 \mathcal{U}_{\varrho} M_2 && \text{iff } \mathfrak{I}, t' \models M_2 \text{ for some } t' \text{ with } t' - t \in \varrho \\
&&& \quad \text{and } \mathfrak{I}, t'' \models M_1 \text{ for all } t'' \in (t, t')
\end{aligned}
$$

了解DatalogMTL扩展运算符

$$\mathit{Authorised}(x) \leftarrow \diamondminus_{[0,2]} \mathit{NegativeLFT}(x) \wedge \boxminus_{[0,15]} \mathit{FullyVaccinated}(x).$$

MeTeoR: metric temporal reasoner

Examples: Fraud Detection

Dataset:

```
RedList(adam)@[0,20]
HighRisk(david)@(-inf,+inf)
HighRisk(ernesto)@(-inf,+inf)
Transaction(adam,betty)@2.87
Transaction(betty,charlie)@12.15
Transaction(charlie,david)@17.5
Transaction(charlie,ernesto)@129.43
```

Query:

```
Suspect(david)@200
```

Program:

```
TransactionChain(X,Y) :- Transaction(X,Y), RedList(X)
TransactionChain(X,Z) :- SOMETIME[-24,0] TransactionChain(X,Y), Transaction(Y,Z)
ALWAYS[0,+inf) Suspect(Y) :-  TransactionChain(X,Y), HighRisk(Y)
```

理解DatalogMTL推理过程，并学习如何利用MeTeoR推理器

进展：理解原生Datalog语言中的MagicSets方法（无MTL）

Input: A Datalog program $\mathcal{P}$, and a query $\mathcal{Q} = g(\bar{t})$.
Output: The optimized program $\mathtt{MS}(\mathcal{Q}, \mathcal{P})$.
var S: **stack** of adorned predicates; *adornedRules*, *modifiedRules*, *magicRules*: **set** of rules;
begin
1. *modifiedRules* := ∅; *magicRules* := ***BuildQuerySeeds***($\mathcal{Q}$, S);
2. **while** $S \neq \emptyset$ **do**
3. p^{α} := S.**pop**( );
4. **for each** rule $r \in \mathcal{P}$ with $H(r) = p(\bar{t}_p)$ **do**
5. *adornedRules* := *adornedRules* ∪ ***Adorn***(r, p^{α}, S);
6. **end while**
7. **for each** rule $r_a \in$ *adornedRules* **do**
8. *magicRules* := *magicRules* ∪ ***Generate***(r_a);
9. **for each** rule $r_a \in$ *adornedRules* **do**
10. *modifiedRules* := *modifiedRules* ∪ {***Modify***(r_a)};
11. $\mathtt{MS}(\mathcal{Q}, \mathcal{P})$:= *magicRules* ∪ *modifiedRules*;
12. **return** $\mathtt{MS}(\mathcal{Q}, \mathcal{P})$;
end.

MagicSets在原生Datalog中应用的伪代码

Program

```
path(X, Y) :- edge(X, Y).
path(X, Y) :- edge(X, Z), path(Z, Y).
```

Query

```
query path(1, 5)
```

MagicSets Transformation (top-down)

Utilizing information from Query to speed up reasoning

```
magic_path^bb(1, 5).
magic_path^bb(Z, Y) :- magic_path^bb(X, Y), edge(X, Z).
path(X, Y) :- magic_path^bb(X, Y), edge(X, Y).
path(X, Y) :- magic_path^bb(X, Y), edge(X, Z), path(Z, Y).
```

Program After MagicSets Method

进展：理解原生Datalog语言中的MagicSets方法（无MTL）

Program

```
path(X, Y) :- edge(X, Y).
path(X, Y) :- edge(X, Z), path(Z, Y).
```

Query

```
query path(1, 5)
```

1 Adornment

```
path_bb(x,y) :- edge(x,y).
path_bb(x,y) :- edge(x,z),path_bb(z,y).
```

2 Generation

```
magic_path_bb(1,5).
magic_path_bb(z,y) :- magic_path_bb(x,y),edge(x,z).
```

During the generation process, some atoms enter the head from the body, and some atoms enter the body from the head.

进展：理解原生Datalog语言中的MagicSets方法（无MTL）

Program

```
path(X, Y) :- edge(X, Y).
path(X, Y) :- edge(X, Z), path(Z, Y).
```

Query

```
query path(1, 5)
```

3 Modification

```
path(x,y) :- magic_path_bb(x,y),edge(x,y).
path(x,y) :- magic_path_bb(x,y),edge(x,z),path(z,y).
```

During the modification process, some atoms enter the body from the head.

Final Program (2 Generation+3 Modification)

```
magic_path^bb(1, 5).
magic_path^bb(Z, Y) :- magic_path^bb(X, Y), edge(X, Z).
path(X, Y) :- magic_path^bb(X, Y), edge(X, Y).
path(X, Y) :- magic_path^bb(X, Y), edge(X, Z), path(Z, Y).
```

进展：尝试将MagicSets方法运用到DatalogMTL中

尝试：假如先把operator and time element丢掉，经过经典Magic Sets转换后再把它们加回去，是否可行?

Dataset:

```
RedList(adam)@[0,20]
HighRisk(david)@(-inf,+inf)
HighRisk(ernesto)@(-inf,+inf)
Transaction(adam,betty)@2.87
Transaction(betty,charlie)@12.15
Transaction(charlie,david)@17.5
Transaction(charlie,ernesto)@129.43
```

Program:

```
TransactionChain(X,Y) :- Transaction(X,Y), RedList(X)
TransactionChain(X,Z) :- SOMETIME[-24,0] TransactionChain(X,Y), Transaction(Y,Z)
ALWAYS[0,+inf) Suspect(Y) :-  TransactionChain(X,Y), HighRisk(Y)
```

Fact:

```
Suspect(david)@200
```

Dataset:

```
RedList(adam)@[0,20]
HighRisk(david)@(-inf,+inf)
HighRisk(ernesto)@(-inf,+inf)
Transaction(adam,betty)@2.87
Transaction(betty,charlie)@12.15
Transaction(charlie,david)@17.5
Transaction(charlie,ernesto)@129.43
magicSuspectb(david)@(-inf,+inf)
```

Program:

```
magicTransactionChainff :- magicSuspectb(X)
ALWAYS[0,+inf) Suspectb(Y) :- magicSuspectb(Y), TransactionChainfb(X,Y), HighRisk(Y)
TransactionChainfb(X,Z) :- magicSuspectb(Z), SOMETIME[-24,0] TransactionChainff(X,Y),
Transaction(Y,Z)
TransactionChainff(X,Z) :- magicTransactionChainff, SOMETIME[-24,0] TransactionChainff(X,Y),
Transaction(Y,Z)
TransactionChainff(X,Y) :- magicTransactionChainff, Transaction(X,Y), RedList(X)
```

Fact:

```
Suspectb(david)@200
```

进展：尝试将MagicSets方法运用到DatalogMTL中

Dataset:	Program:	Fact:
RedList(adam)@[0,20] HighRisk(david)@(-inf,+inf) HighRisk(ernesto)@(-inf,+inf) Transaction(adam,betty)@2.87 Transaction(betty,charlie)@12.15 Transaction(charlie,david)@17.5 Transaction(charlie,ernesto)@129.43 magicSuspectb(david)@(-inf,+inf)	magicTransactionChainff :- magicSuspectb(X) ALWAYS[0,+inf) Suspectb(Y) :- magicSuspectb(Y), TransactionChainfb(X,Y), HighRisk(Y) TransactionChainfb(X,Z) :- magicSuspectb(Z), SOMETIME[-24,0] TransactionChainff(X,Y), Transaction(Y,Z) TransactionChainff(X,Z) :- magicTransactionChainff, SOMETIME[-24,0] TransactionChainff(X,Y), Transaction(Y,Z) TransactionChainff(X,Y) :- magicTransactionChainff, Transaction(X,Y), RedList(X)	Suspectb(david)@200

出现问题：

1、我们将从fact得到的magic规则的time element设定为无穷 (-inf, +inf)，**能利用fact的term内容对推理范围进行top-down的过滤**。但是这种方法并不严谨，而且query中的时间信息尚未利用。我们如何将magic fact的time element设定成与原来的fact相同，**进一步利用query中的时间信息对推理范围进行过滤?**

2、在语法上，**只有ALWAYS这个operator被允许出现在规则的head中（定义）**。但在generation过程中，所有类型的operator and time element都可能从body进入到head中。此时的SOMETIME、UNTIL是应当被舍弃，还是可以转换成ALWAYS？无论如何，在进行MagicSets变换时，**需要对operator and time element进行处理**。

成果：尝试将MagicSets方法运用到DatalogMTL中

探索：针对operator and time element从head进入body和从body进入head这两种情况，对operator and time element进行处理。使得规则能推理出正确的结果，同时符合规则语法的要求。

atom所带的operator	Generation从head进入body时	Generation从body进入head时	Modification从head进入body时
ALWAYS	变为SOMETIME	保留ALWAYS不变	变为SOMETIME
SOMETIME	/	变为ALWAYS	/
UNTIL	/	变为两条ALWAYS开头的规则（未验证）	/

原理：对时间限制的“放宽”

是否会导致多余的事实被推导出来？：不会，因为影响的其实只有magic事实，而magic事实只相当于起到过滤器的作用，把程序的evaluation限制在可能推导出查询结果的范围内。

成果：尝试将MagicSets方法运用到DatalogMTL中

Dataset:

```
Infected(a)@[0,10]
Meet(a,b)@9
Meet(b,a)@9
Meet(b,c)@[26,28]
Meet(c,b)@[26,28]
Meet(c,d)@45
Meet(d,c)@45
```

Program:

```
ALWAYS[10,20] Infected(X) :-  Meet(X,Y), Infected(Y)
```

Fact:

```
Infected(d)@56
```

不对ALWAYS进行处理：
推理失败

Program:

```
magic_Infected_b(Y) :- ALWAYS[10,20] magic_Infected_b(X), Meet(X,Y)
ALWAYS[10,20]Infected(X) :-  magic_Infected_b(X),Meet(X,Y),Infected(Y)
```

Run to check if the fact is entailed by the dataset and the program

Answer: Not Entailed

对ALWAYS进行处理：推理成功，
且得到的materialized facts与
bottom-up方法得到的完全符合

Program:

```
magic_Infected_b(Y) :- SOMETIME[10,20] magic_Infected_b(X), Meet(X,Y)
ALWAYS[10,20]Infected(X) :-  SOMETIME[10,20] magic_Infected_b(X),Meet(X,Y),Infected(Y)
```

Run to check if the fact is entailed by the dataset and the program

Answer: Entailed

成果：编写MagicSets方法在DatalogMTL中实现的伪代码

Algorithm 1: Optimize Datalog Program

Input : A Datalog program P, and a query $Q = g(\bar{t})$
Output: The optimized program $MS(Q, P)$

$stack \leftarrow \emptyset$; $adornedRules \leftarrow \emptyset$; $modifiedRules \leftarrow \emptyset$;
$magicRules \leftarrow$ **BuildQuerySeeds**$(Q, stack)$;
while $stack \neq \emptyset$ **do**
  $p^{\alpha} \leftarrow stack$.pop();
  foreach *rule* $r \in P$ *with* $H(r) = p(\bar{t}_p)$ **do**
    $adornedRules \leftarrow adornedRules \cup$ **Adorn**$(r, p^{\alpha}, stack)$;
foreach *rule* $r_a \in adornedRules$ **do**
  foreach *adorned atom* $b \in B(r_a)$ **do**
    $H(r_m) \leftarrow$ **MagicBodyToHead**(b) ;
    $B(r_m) \leftarrow$ **MagicHeadToBody**$(H(r_a))$ followed by all atoms $\in$ $\{a | a \in B(r_a), a$ is ahead of $b\}$;
    $magicRules \leftarrow magicRules \cup \{r_m\}$;
foreach *rule* $r_a \in adornedRules$ **do**
  Insert **MagicHeadToBody**$(H(r_a))$ to $B(r_a)$;
  Strip off the adornments of all other predicates in r_a ;
  $magicRules \leftarrow magicRules \cup \{r_a\}$;
$MS(Q, P) \leftarrow magicRules \cup modifiedRules$;
return $MS(Q, P)$;

Algorithm 1: MagicSets Transformation in DatalogMTL

Algorithm 2: MagicHeadToBody Function

Input : An atom $a = p^{\alpha}(\overline{v})$
Output: An atom a'

$\overline{v}' \leftarrow$ eliminate all arguments corresponding to an f label in α in $\overline{v}$;
$magic\_p^{\alpha} \leftarrow$ attach the prefix "magic_" to the predicate symbol p^{α};
$a' \leftarrow magic\_p^{\alpha}(\overline{v}')$;
if $a' = always_{\varrho}M$ **then**
  return $sometime_{\varrho}M$
else
  return a'

Algorithm 2: MagicHeadToBody Function

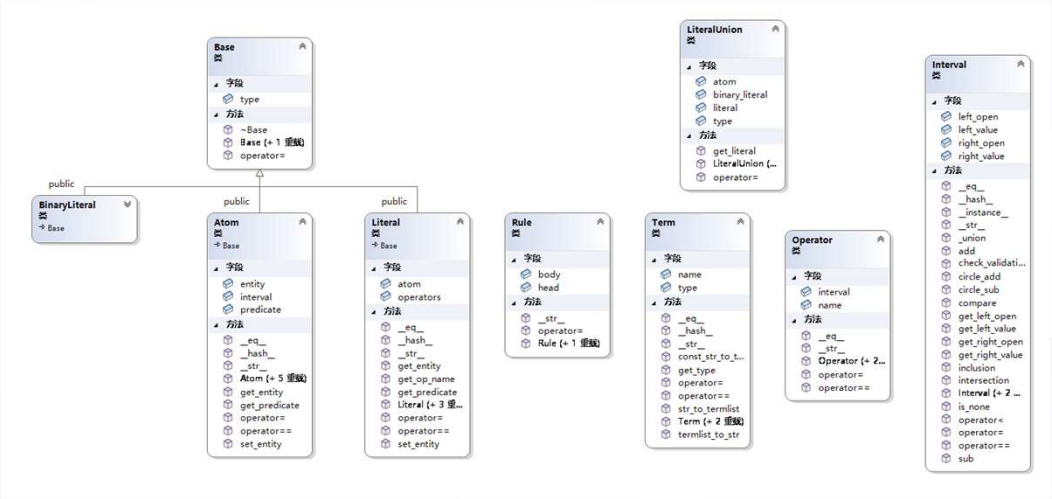
Algorithm 3: MagicBodyToHead Function

Input : An atom $a = p^{\alpha}(\overline{v})$
Output: An atom a'

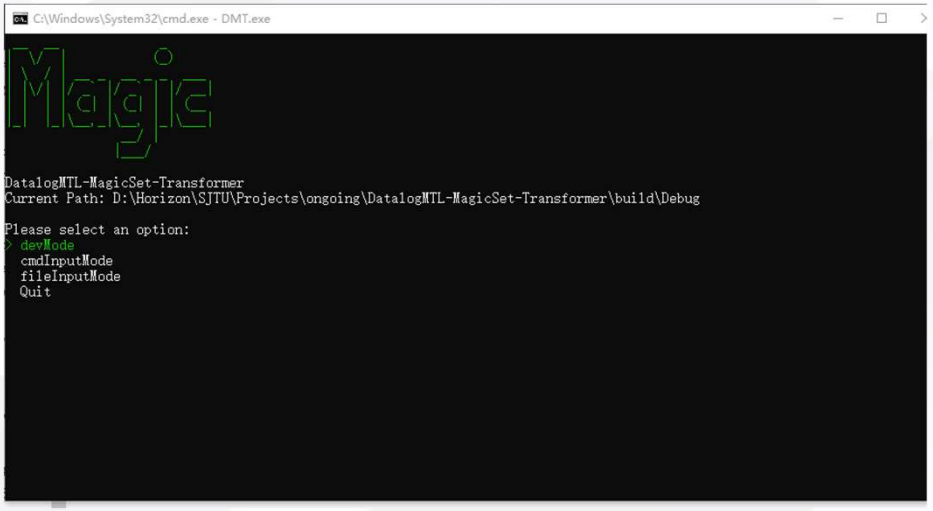
$\overline{v}' \leftarrow$ eliminate all arguments corresponding to an f label in α in $\overline{v}$;
$magic\_p^{\alpha} \leftarrow$ attach the prefix "magic_" to the predicate symbol p^{α};
$a' \leftarrow magic\_p^{\alpha}(\overline{v}')$;
if $a' = sometime_{\varrho}M$ **then**
  return $always_{\varrho}M$
else if $a' = M_1 until_{\varrho} M_2$ **then**
  return ...
else
  return a'

Algorithm 3: MagicBodyToHead Function

成果：完成了代码的实现



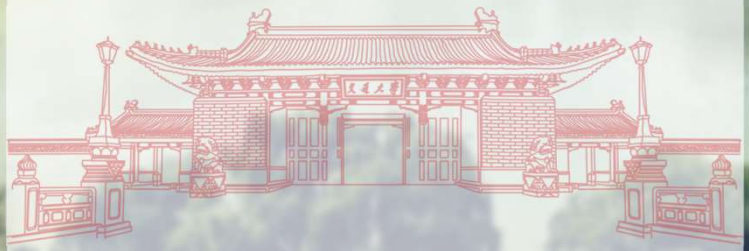
项目的类图



实际运行展示

02

问题与解决思路



问题与解决思路

问题一：对于DatalogMTL的四种时间相关的运算符，在使用原生MagicSets方法进行转化时遇到困难。

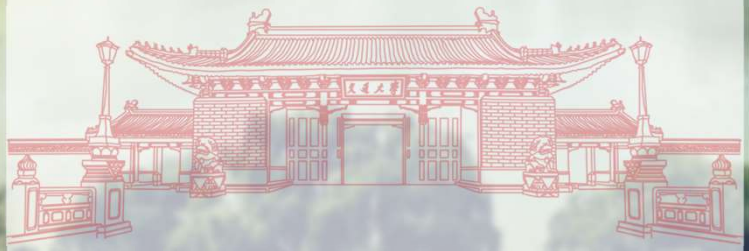
解决思路一：针对这个问题，我们决定先研究其中的Sometimes和Always运算符，因为它们作为一元运算符，可以直接基于原生MagicSets方法，再加以修饰后进行可靠的扩展。而对于Since和Until这两个作为二元运算符的情况，我们决定将其放在之后的日程中继续深入研究。

问题二：对于MagicSets方法在DatalogMTL中的严谨证明方面遇到了困难。

解决思路二：针对这个问题，我们通过和指导老师的探讨和研究，正在从Principles of Database and Knowledge-Base Systems Volume II中学习可参考的证明方法。通过不断的交流和学习，我们期望能够更好地理解并严谨证明DatalogMTL中的MagicSets方法，从而顺利解决这一困难。

03

下阶段主要安排



下阶段主要安排

主要任务：

1、严谨证明DatalogMTL中的MagicSets方法

2、进一步完善MagicSets方法的C++实现（基于严谨证明和二元MTL运算符的研究）

3、对该方法在不同数据集下的性能和效果进行测试。

具体时间安排为：

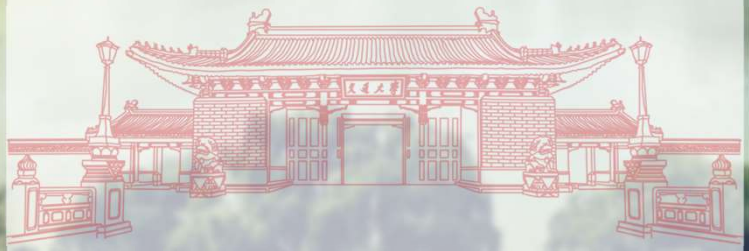
2024年3月-2024年7月：严谨证明DatalogMTL中的MagicSets方法并完善代码

2024年8月-2024年10月：在不同数据集下的性能和效果进行测试

2024年11月-2025年3月：前述算法的集成、引擎优化和论文整理投稿

04

科研体会和心得



科研体会和心得

1、深刻认识到研究的创新性和挑战性

在项目进行过程中，团队成员深刻认识到MagicSets方法在DatalogMTL中的**创新性**和**挑战性**。通过不断的学习和探索，团队成员增长了对于MagicSets方法在DatalogMTL推理中的理解，也提升了解决相关问题的能力和实践经验。

2、体会到了创新思维和创新实践的重要性

在项目实施过程中，通过不断的思考和尝试，我们找出了DatalogMTL推理优化的切入点，并提出了将MagicSets方法应用到DatalogMTL中的创新思路。在实践中，团队成员充分发挥自己的想象力和创造力，不断尝试各种可能性，将情况初步确定在Sometimes和Always运算符在MTL语句中的处理，从而取得了初步的成效。因此，我们也深刻体会到了**创新思维**和**创新实践**的重要性。

3、展望

在未来的项目实施中，团队将继续保持创新思维，不断开拓创新，不断进行实践，为项目的成功实施和更多成果的取得努力奋斗。

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

感谢老师！

饮 水 思 源　爱 国 荣 校