Dist-AI in TLA+*

Xiaosong Gu
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
xxx@smail.nju.edu.cn

Jiacheng Zhao
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China

Wenjun Cai
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
xxx@smail.nju.edu.cn

Hengfeng Wei*
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
hfwei@nju.edu.cn

Yu Huang
State Key Laboratory for Novel Software Technology
Nanjing University
Nanjing, China
yuhuang@nju.edu.cn

. . .

. . .

摘要

PVLDB Reference Format:

Xiaosong Gu, Jiacheng Zhao, Wenjun Cai, Hengfeng Wei*, Yu Huang, . . . , and . . . . Dist-AI in TLA+. PVLDB, 14(1): XXX-XXX, 2020.

doi:XX.XX/XXX.XX

PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at URL_TO_YOUR_ARTIFACTS.

*Corresponding author. Hengfeng Wei is also with Software Institute at Nanjing University.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 INTRODUCTION

TLA+, TLC, and TLAPS.

Automatic invariant inference.

Overview.

- **TLA+traces sampling**
  - **Counter-example Guided**
  - **Coverage (e.g., minimal spanning)**
- **invariants space enumeration (exploration)**
  - **using Apalache: VARIABLES to relations (in Ivy), which are used as items in invariants**
  - **convert invariants in terms of relations back to those in terms of TLA+ variables**
- **Validation (utilizing Apalache)**
  - **on finite models; for any steps**
- **Refinement**
  - **Counter-example Guided**
- **Generalization to any models (for any steps)**
  - **How to validate it? (find some SMT???)**

Our Contributions.

-
-
-

2 OVERVIEW

2.1 流程

- 通过类型分析，得到TLA+ 规约中的变量类型
  Apalache工具, Json格式
- 根据转换规则，将每个变量用一组**relation**表示
- 对TLA+的**trace**，将每个状态中变量的值替换为**relation**的值，得到用关系表示的**trace**
- 将用关系表示的**trace**发送给**DistAI**，得到候选的不变式
  DistAI只能生成没有∃的公式
  由于**TLA+**的变量的数据结构的复杂性，每个变量可能需要一组关系才能表示，这样的关系可能整体考虑才有意义，但是**DistAI** 的枚举没有这样的保证
  用关系表示时会引入局部变量
- 将由**relation**组成的不变式中的**relation**替换成**TLA+**中变量的符号，得到用**TLA+**变量组成的候选不变式
- 将候选的不便式通过**Apalache**通过检验

2.2 转换规则

TLA+ 的类型系统定义:

$t \doteq Cons \mid Bool \mid Int \mid t \to t \mid Set(t) \mid t \times \cdots \times t \mid [nm_1 : t, \cdots, nm_k : t]$

将每一个变量v,用一组关系RS_v表示:

- 若$v.type = Cons \mid Int \mid Bool$, 定义关系$R_v(_) : v = x \leftrightarrow R_v(x)$. 则$RS_v = R_v$.
- 若$v.type = Set(t)$,定义$R_v(_) : x \in v \leftrightarrow R_v(x)$
  若$t \in tb$, 则$RS_v = R_v$
  若$t \notin tb$, 则$RS_v = R_v(x) \wedge RS_x$($x$是$t$类型的一个占位符)
- 若$v.type = T_1 \times T_2 \times \cdots \times T_k$, 则定义$k$个关系$R_i(_) : v[i] = x \leftrightarrow R_i(x)$。对每个关系$R_i$, 定义$RS_i$:
  若$T_i \in tb$, 则$RS_i = R_i$
  若$T_i \notin tb$, 则$RS_i = R_i(x) \wedge RS_x$($x$是$T_i$类型的占位符)
  则$RS_v = RS_1 \wedge RS_2 \wedge \cdots \wedge RS_k$
- 若$v.type = T_1 \to T_2 \to \cdots \to T_k$, 则定义$(k+1)$元关系 $R_v(\_, \cdots \_) : v(x_1, \cdots, x_k) = x_{k+1} \equiv R_v(x_1, \cdots, x_k.x_{k+1})$。其中,
  若$T_i \in \{Cons, Int, Bool\}$, 则$x_i$为具体的值, 且$RS_i = true$
  若$T_i \notin tb$, 则x_i为T_i类型的占位符, 且$RS_i = RS_{x_i}$
  则$RS_v = R_v(x_1, \cdots, x_k, x_{k+1}) \wedge RS_{x_1} \wedge RS_{x_2} \wedge \cdots \wedge RS_{x_{k+1}}$
- 若$v.type = [nm_1 : T_1, \cdots, nm_k : T_k]$, 由于**record**类型的各个项之间没有顺序关系, 所以可以按照某种规则(如字典序) 将$\{nm_1, \cdots, nm_k\}$ 进行排序. 假设已经进行了排序, 那么**record**类型的规则和k元**tuple**相同。定义k个关系$R_i(_)$: $v.nm_k = x \leftrightarrow R_k(x)$. 对于每个关系$R_i$, 定义$RS_i$:
  若$T_i \in \{Int, Bool, Cons\}$, 则$RS_i = R_i$
  若$T_i \notin \{Int, Bool, Cons\}$, ß$RS_i = R_i(x) \wedge RS_x$
  则$RS_v = RS_1 \wedge RS_2 \wedge \cdots \wedge RS_k$

2.3 Sampling TLA+ Traces

2.4 Enumerating Invariants

- **directed by syntax of TLA+**
- **restricting terms, operations, . . .**

2.5 Validating Inductive Invariants

- **using Apalache (modified for validating fols with quantifiers)**
- **using [?]**

表 1: Details of the TPCommit Variables

Name	Type
`rmState`	(Str -> Str)
`tmState`	Str
`tmPrepared`	Set(Str)
`msgs`	Set([rm: Str, type: Str])

3 CASE STUDY

3.1 Lock Server

3.2 Two-Phase Commit

We use Two-Phase Commit as an example. First, we need to extract the main variables of this protocol. Their types are listed as follows. They are marked on the top of the TLA+ code, so when the type check work is done, they can easily be extracted.

The next step is to calculate the possible values of these variables, and to generate relations for them. Variable `rmState`, `tmState` and `tmPrepared` have clear value ranges, which have been listed in the `TPTypeOK` section. Some of these relations are listed below.

`tmState(x) : tmState = x`, in which `x` can be either`"init"`, `"commited"` or `"aborted"`

`rmState(x,y) : rmState(x) = y`, in which `x` can be all RMs, `y` can be the four strings specified by `TPTypeOK` section.

`tmPrepared(x) : x ∈ tmPrepared`, in which `x` can be all RMs.

Relation generated from `msgs` is relatively difficult. First we need to calculate the maximum number of elements in `msgs`, which is 3.(The way to calculate this number is still unknown.) Each possible element (or record) in `msgs` contains two properties, which are `type` and `rm`. Thus we need to define seven relations as below.

```
msgs(x) : x ∈ msgs
msgs.msg1.type(y) : x.type = y
msgs.msg1.rm(y) : x.rm = y
msgs.msg2.type(y) : x.type = y
msgs.msg2.rm(y) : x.rm = y
msgs.msg3.type(y) : x.type = y
msgs.msg3.rm(y) : x.rm = y
```

It's worth noting that all the `x` occurred in these relations are unnamed variables. In practice they are given random names without repetition, such as `msgsv1`, to indicate that it is the first element of the set `msgs`. Using `msgsv1` here doesn't mean that we are sorting elements in a set, instead it is only a way to maintain the consistency among all these three relations. We are able to depict the states of `msgs` with these three relations.

10 relations are defined above, and 31 predicates will be generated assuming that RM is a 2-element set. DistAI will use these predicates to handle further processes.

There is another way to depict `msgs`. The six relations used to describe elements within msgs can be synthesized. Thus we can replace the 8 relations with the 2 relations below. This method reduces the amount of relations, but increases the amount of predicates. In the 2-element case, we would have 5 relations and 34 predicates.

```
msgs(x) : x ∈ msgs
msgs.type(x,y,z) : x.rm = y ∧ x.type = z
```

3.3 Paxos

关于Paxos规约中的例子:

- 变量$PrepareMsg : Node \rightarrow Ballot \rightarrow \langle Ballot, Value \rangle$. 那么$\forall s \in Node, a, b \in Ballot, v \in Value : PrepareMsg[s][a] = \langle b, v \rangle \leftrightarrow R_p(s, a, tup) \wedge R_{tup1}(b) \wedge R_{tup2}(v)$ (tup是类型$Ballot \times Value$的一个符号)
- 变量$Decision : Node \rightarrow Ballot \rightarrow Set(Value)$, 那么$\forall s \in Node, a \in Ballot, v \in Value : v \in$

$Decision[s][a] \leftrightarrow R(s, a, set) \wedge R_{set}(v)$ **(**set**是类型**$Set(Value)$**的一个符号)**

- **若定义**$Messages = [type : \{$**“prepare”**, **“aborted”**$\}, svr : Server]$**, 变量**$msgs \subseteq Messages$**. 那么若**$\forall tp \in \{$**“prepare”**, **“aborted”**$\}, s \in Server : [tp, s] \in msgs \leftrightarrow R_(m) \wedge R\_m\_1(tp) \wedge R\_m\_2(s)$ **(**m**是**$Message$**类型的一个符号)**
- $Messages = [type : \{$**“prepare”**, **“aborted”**$\}, svr : Server] \cup [type : \{$**“commit”**$\}]$**, 不能直接转换.**

4 RELATED WORK

DistAI

SWISS

Ivy

I4: inductive invariants for finite models (utilizing Averroes), and then generalize them to general models

Apalache

5 CONCLUSION

@inproceedingsProofAutomation:PhDThesis2014, title=Proof automation and type synthesis for set theory in the context of TLA+, author=Hernán Vanzetto, year=2014