Semantics

Roland Meyer

TU Braunschweig

Table of Contents I

1. Type-based Algorithm due to Kobayashi and Ong
   - KO Types
   - KO Type Environment and Type Judgements

Part C Higher-Order Model Checking

1. Type-based Algorithm due to Kobayashi and Ong

Type-based Algorithm

Goal

Give a type-based algorithm for solving HOMC.

Given APTA A, construct type system $\mathcal{T}_A$.

Want following equivalence, for every scheme G:

$$G \text{ type checks in } \mathcal{T}_A \qquad \text{iff} \qquad [\![G]\!] \text{ is accepted by } A \ .$$

Background

Result due to Kobayashi and Ong, LICS'09.

Type-based Algorithm

Advantages of Kobayashi and Ong'09 over Ong'06

Simplicity

Correctness follows from two arguments:

- Correctness of the type system.
- Correctness of the type-checking algorithm.

Complexity

If the automaton is fixed and

the size of types is bounded by a constant

- then the algorithm runs in time linear in the size of the scheme.

Ong'06 still needs n-EXPTIME

Flexibility

Algorithm can be modified to deal with extensions of schemes:

- Polymorphism $(o \rightarrow o) \wedge ((o \rightarrow o) \rightarrow (o \rightarrow o))$
- Finite data domains

Type-based Algorithm

Technology

From type-theoretic point of view,

type system has interesting new features:

- Flags and priorities express when a variable can be used.
- Type check amounts to determining the winner in a parity game.

1.1. KO Types

KO Types

Definition

Let $A = (Q, \Sigma, \delta, q_{init}, \Omega)$.

The set of atomic types θ and the set of types τ

are defined by simultaneous induction:

$$\theta \ ::= \ q \mid \underbrace{\tau}_{\text{type}} \to \theta$$

$$\tau \ ::= \ \bigwedge\{(\underbrace{\theta_1}_{\text{atomic type}}, m_1), \ldots, (\theta_k, m_k)\} \ .$$

Here, $q \in Q$ and $m_1, \ldots, m_k \in \mathit{range}(\Omega)$.

KO Types

Notation

Write $\bigwedge\{(\theta_1, m_1), \ldots, (\theta_k, m_k)\}$ as

$$(\theta_1, m_1) \wedge \ldots \wedge (\theta_k, m_k) \qquad \text{or} \qquad \bigwedge_{i=1}^{k} (\theta_i, m_i)\ .$$

Write $\top$ for $\bigwedge \emptyset$.

Currently have priorities only for states.

Extend this to all atomic types:

$$\Omega(\tau \to \theta) \ := \ \Omega(\theta)\ .$$

KO Types

Intuition

Type $(q_1, m_1) \wedge \ldots \wedge (q_k, m_k) \to q$ describes a function that

takes a tree accepted from q_1 and from q_2 and ... from q_k and

returns a tree that is accepted from q.

Priority m_i describes the maximal priority on the path

from the root of the output tree (of type q)

to the root of the input tree (of type q_i).

Consequence

The input tree can be used as a tree of type q_i only

after visiting a state of priority m_i and

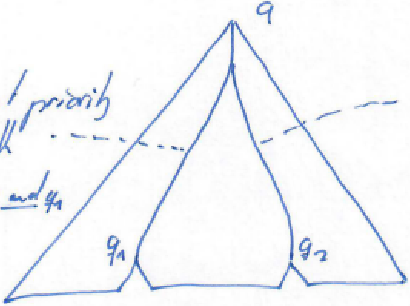
before visiting a state of priority $> m_i$.

KO Types

Illustration

The type $(q_1, m_1) \wedge (q_2, m_2) \to q$
describes the following tree function:

The largest priority on this path including q and q_1 is m_1.



The largest priority including q and q_2 is m_2.

KO Types

So far, types are not related to kinds.

Define well-formed types via two relations:

$$\tau :: k \;=\; \tau \text{ is a type of kind } k$$

$$\theta ::_a k \;=\; \theta \text{ is an atomic type of kind } k\,.$$

Definition

The relations $::$ and $::_a$ are the least relations

satisfying the following rules:

$$\frac{}{q ::_a o} \qquad \frac{\tau :: k_1 \quad \theta ::_a k_2}{\tau \to \theta ::_a k_1 \to k_2} \qquad \frac{\theta_i ::_a k \quad \text{for all } 1 \leq i \leq l}{\bigwedge\{(\theta_1, m_1), \ldots, (\theta_l, m_l)\} :: k}\,.$$

KO Types

Definition

A type τ and an atomic type θ are well-formed, if

(1) $\tau :: k$ resp. $\theta ::_a k$ for some kind k and

(2) for each subexpression $\bigwedge_{i=1}^{l}(\theta_i, m_i) \to \theta'$ we have

$$m_i \;\geq\; \max\{\Omega(\theta_i), \Omega(\theta')\} \quad \text{for all } 1 \leq i \leq l\,.$$

Example

- $q_1 \wedge ((q_2, 1) \to q_3)$ is not well-formed.

It combines types of different kinds, and hence (1) fails.

- $(q_1, m_1) \wedge (q_2, m_2) \to q)$ is well-formed, provided

$$m_1 \;\geq\; \max\{\Omega(q_1), \Omega(q)\} \qquad \text{and} \qquad m_2 \;\geq\; \max\{\Omega(q_2), \Omega(q)\}\,.$$

This reflects the fact that m_1 and m_2 are the largest priorities on the above paths, including the root and the leaves.

From now on, only consider well-formed types.

1.2. KO Type Environment and Type Judgements

KO Type Environment and Type Judgements

Definition

A flagged type is an expression $(\theta, m)^b$ with $b \in \mathbb{B} = \{\mathit{true}, \mathit{false}\}$.

We use σ for flagged types.

A type environment Γ is a set of bindings $x : \sigma$.

With this definition, type judgements will have the form

$$\Gamma \vdash t : \theta \ .$$

Here, t will be a term.

We will treat its non-terminals as variables that are bound by Γ.

Note that Γ uses flagged types.

Term t, however, receives a normal (well-typed) atomic type.

KO Type Environment and Type Judgements

Explanation

- Γ may contain several bindings for the same variable.

- Each atomic type of a variable is annotated by a flag.

The flag indicates when the variable can be used

as a value of that type:

1. $x : (\theta, m)^{true} \in \Gamma$

   means x can only be used

   before visiting a state of priority $> m$.

2. $x : (\theta, m)^{false} \in \Gamma$

   means x can only be used

   before visiting a state of priority $> m$ and

   after visiting a state of priority $= m$.

   Hence, if $x : (\theta, m)^{false} \in \Gamma$, then the largest priority

   on the path from the current node to the node where x is used, equals m.

KO Type Environment and Type Judgements

We have not yet defined the set of type judgements
(that we consider valid).

The following examples (on the board) are meant to give some intuition
to which type judgements should be valid.