

HCVS2023 notification for paper 9688

HCVS2023 <hcv2023@easychair.org>

Sun 02/04/2023 12:39

To: Bajczi Levente <levente.bajczi@edu.bme.hu>

Dear Levente,

We are pleased to inform you that your paper has been selected for presentation at HCVS 2023. Instructions for preparing the EPTCS proceedings will be sent in a separate email.

SUBMISSION: 9688

TITLE: Bottoms Up for CHCs: Novel Transformation of Linear Constrained Horn Clauses to Software Verification

----- REVIEW 1 -----

SUBMISSION: 9688

TITLE: Bottoms Up for CHCs: Novel Transformation of Linear Constrained Horn Clauses to Software Verification

AUTHORS: Márk Somorjai, Mihály Dobos-Kovács, Zsófia Ádám, Levente Bajczi and András Vörös

----- Overall evaluation -----

SCORE: 1 (weak accept)

----- TEXT:

The paper presents a method of transforming constraint Horn Clauses into Control Flow Automata; and then leveraging facilities of a model-checking tool to decide satisfiability.

The method is called "forward", as it assembles the Automata iteratively starting with simple atomic facts.

The paper reads well, and the running example serves the purpose of explaining concretely how the general method works. (Modest) evaluation is given, and shows that the presented method can be effective.

The paper is well on the topic of HCVS, and should be accepted.

A query: the forward and backward terminology reminded me of the standard Horn Clause Theory dichotomy, between bottom up assembly of Herbrand models from facts; and topdown (SLD)-resolution search. It might be worthy of contrasting the intuitions arising from this paper with these classical notions.

----- REVIEW 2 -----

SUBMISSION: 9688

TITLE: Bottoms Up for CHCs: Novel Transformation of Linear Constrained Horn Clauses to Software Verification

AUTHORS: Márk Somorjai, Mihály Dobos-Kovács, Zsófia Ádám, Levente Bajczi and András Vörös

----- Overall evaluation -----

SCORE: 1 (weak accept)

----- TEXT:

Summary:

A translation from linear CHCs to CFA (Control-Flow Automata) is presented in this paper. The goal is to allow the CFA-based model checker THETA to compare with existing CHC solvers using the benchmarks from CHC-COMP.

The translation is straightforward. Predicates are translated to CFA locations. CHC constraints are translated to a sequence of statements. Special care is needed with recursive CHCs. In addition, the paper explain how to map invariants and counterexamples to CHC models and refutations.

The evaluation has two variants: one using a forward transformation from CHCs to CFA and the other using a backward transformation. The evaluation on LIA-Lin benchmarks shows that THETA with forward CHC transformation behaves close to Eldarica although a bit worse. This result is not surprising since Eldarica implements a similar verification algorithm (CEGAR using predicate abstraction)

Evaluation:

The paper describes a CHC front-end for an existing model-checker called THETA.

Pros:

- The community can benefit from having another CHC solver.

Cons:

- The front-end does not present any novel idea.
- The evaluation does not provide any new insights.
- The poor results of backward THETA are not explained.

Although I don't see any technical contribution, I give credit for implementing a CHC front-end that allows comparing model-checkers with existing CHC solvers. The topic of the paper is definitely a good fit for the workshop.

Questions:

- What is the backward transformation from CHC to CFA? This needs to be clarified in the final version. Does it mean that running THETA on the CFA has a similar effect to backward CHC solvers? If my guess is right then how the backward transformation compares with magic-set transformation [1]?
- Related to previous question. Do you have an intuition why backward THETA is much worse than forward THETA? I'm surprised about that.
- Regarding the translation from CHC constraints to statements. I think several important details are missing. Not all equalities can be translated to assignments. In fact, a straightforward translation would do the opposite: translate all equalities into assumes/conditionals. However, the examples in the paper translate into assignments. How do you know when to translate to an assignment? Also, I assume that CHCs must be first preprocessed to make the translation simpler. For instance, I would transform all CHCs into Negation Normal Form. Are you doing something similar? CHC constraints have many or and ite terms. How are they translated? Finally, can the front-end support other theories beyond LIA? For instance, arrays?

Could you please elaborate a bit more about all these details?

More comments:

This work would benefit if the translation is public and it's designed in such way that it would be relatively easy for other CFA-based model checkers to benefit from it (e.g., CPAchecker)

I would also argue that the title of the paper is a bit overselling.

References:

[1] magic-set transformation is an idea from the 80s but I couldn't find the original reference. <https://souffle-lang.github.io/magicset>.

