

Abstract topology analysis of the join phase of the merge protocol^{*}

Peter Backes¹ and Jan Reineke²

¹ Universität des Saarlandes, Saarbrücken, Germany
`rtc@cs.uni-sb.de`

² University of California, Berkeley
`reineke@eecs.berkeley.edu`

Abstract. We present a partial solution to the TTC2010 topology analysis case study. We pick a small part of the merge protocol, namely the part where cars join a leader to form a platoon. Using abstract interpretation, we compute an approximation of the arising topologies, without limiting the number of cars.

1 Introduction

In our case study, we ask “Can you analyze the protocol in a general way using abstraction techniques such that the number of processes is not limited?” In this solution to the case study, we achieve this goal for a part of the protocol. Section 2.2 of the case study mentions that the protocol implements three tasks. The part that we analyze consists of the first and the second task: Building a platoon out of two processes so that one of them becomes a leader and the other a follower, and having a process join an existing platoon.

If we apply the merge protocol graph transformation rules from the case study to a start graph with finitely many nodes, then they produce a finite set of graphs. This is so because none of the rules adds new nodes. If, on the other hand, we use an empty start graph, and add a new rule that merely creates free agent nodes, the result will be an infinite set of graphs. Accordingly, such a system cannot be analyzed using classic graph transformation tools.

Abstract interpretation allows us to compute approximations of such systems. The state of the art technique using this paradigm is partner abstraction [2,3], implemented in the tool `hiralysis`. However, partner abstraction was designed

^{*} This work was supported by the DFG as part of the Transregional Collaborative Research Center SFB/TR 14 AVACS. In addition, it was supported in part by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, which receives support from the National Science Foundation (NSF awards #0720882 (CSR-EHS: PRET) and #0931843 (ActionWebs), the U. S. Army Research Office (ARO #W911NF-07-2-0019), the U. S. Air Force Office of Scientific Research (MURI #FA9550-06-0312 and AF-TRUST #FA9550-06-1-0244), the Air Force Research Lab (AFRL), the Multiscale Systems Center (MuSyC) and the following companies: Bosch, National Instruments, Thales, and Toyota.

for simple topologies only and requires a human expert to supply additional invariants—partner constraints—to cut off parts of the merge protocol that involve more complicated topologies. Such more complicated topologies do occur in the merge protocol [1]. Without cutting off these parts, partner abstraction will suffer from state space explosion, and `hiralysis` will not terminate.

The more complicated topologies cut off for partner abstraction analysis already arise for the two mentioned tasks. We aim at an abstraction that does not need manual intervention to cope with these topologies.

In section 2, we introduce our abstraction. Section 3 presents which parts of the protocol we analyze and the abstract result that we get after running `astra`, our analysis tool. We talk about the evaluation of properties on the abstract result in section 4. Section 5 sums up our work and presents future research.

2 Star abstraction

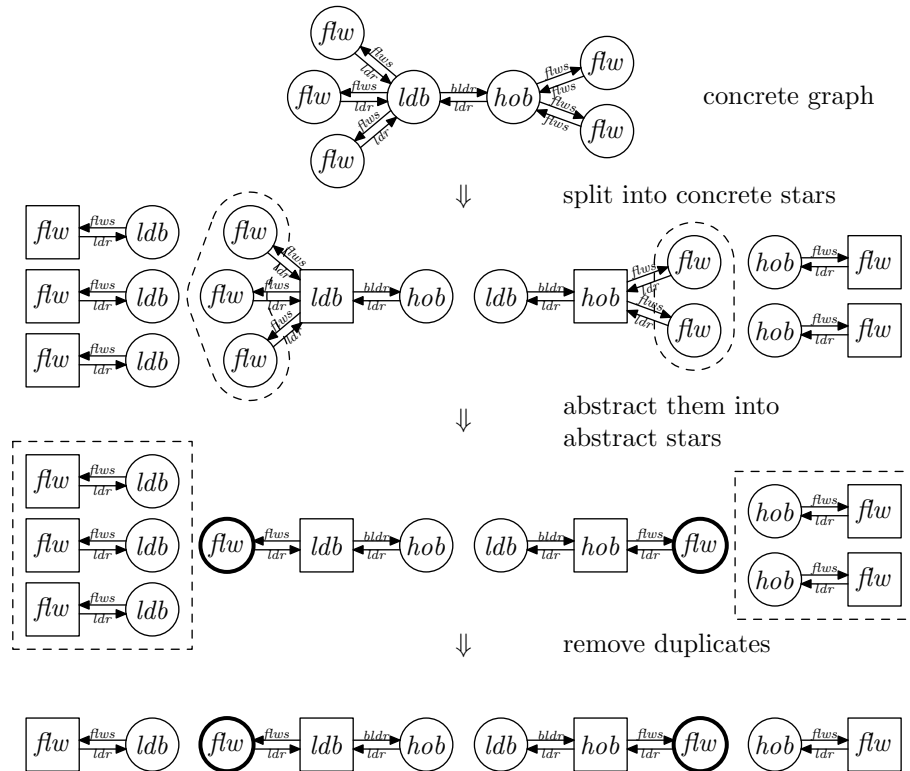


Fig. 1. Our abstraction applied step by step to a simple example

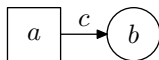
Our abstraction is sketched in Figure 1. We abstract graphs in three steps: First, we build the corresponding *star* for all nodes v of the graph. We obtain the star by removing all nodes from the graph except for v and its partner nodes, and by removing all the edges that are not incident to v . We call v the *core node* (displayed as a square in Figure 1) and the other nodes the *outer nodes*.

The next step is done for each star separately. We identify sets of outer nodes that cannot be distinguished from each other with respect to their label and the labels and directions of the edges incident to them. For each such set that contains two or more indistinguishable nodes, we merge them all into a summary node. We are then left with *abstract stars*. We represent the abstract stars as a tuple (l, E, A, S) with $l \in \mathcal{N}$ being the label of the core node, $E \subseteq \mathcal{E}$ being the self-loops of the core node, $A \subseteq \mathcal{N} \times 2^{\mathcal{E}} \times 2^{\mathcal{E}}$ being the *axes* and $S \subseteq A$ specifying which of those axes have summary nodes. Each axis (l, in, out) represents an outer node with label l and the edges incident to it. *in* contains those edges that point from the core node to the outer node and *out* the remaining ones. By construction, it follows that at least one connection must exist, that is, $in \cup out \neq \emptyset$.

In the final step, we ensure that each abstract star is unique. This is accomplished by keeping at most one copy from each class of isomorphic abstract stars.

There are $|\mathcal{N}| \cdot 2^{|\mathcal{E}|} \cdot 3^{|\mathcal{N}| \cdot (2^{2 \cdot |\mathcal{E}|} - 1)}$ different stars: Each star can have $|\mathcal{N}|$ different node labels for the core node and can have any subset of the $|\mathcal{E}|$ edge labels for self-loops. Between the core node and each outer node, there can be any edge with one of the $|\mathcal{E}|$ edge labels, either pointing from the core node to the outer node, or the other way around; with the exception that in total, at least one edge must be present. The outer node of an axis can have any of the $|\mathcal{N}|$ node labels, and each axis is either present, absent or present as a summary axis. We denote the set of all possible stars over a node label set \mathcal{N} and an edge label set \mathcal{E} as $\mathfrak{S}(\mathcal{N}, \mathcal{E})$.

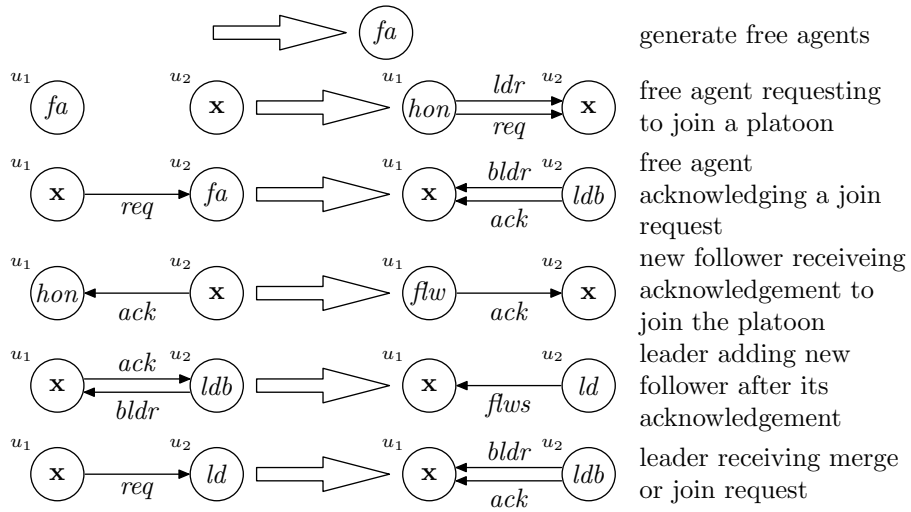
We do not yet have a result on the exact size of the star domain itself and merely know that it is bounded by $2^{|\mathfrak{S}(\mathcal{N}, \mathcal{E})|}$. It is non-trivial, because not every subset of the stars is a valid abstraction of an actually existing concrete graph. For example, a set containing only the star



is not a abstraction of any graph, since there is no corresponding star with a core node that has label b .

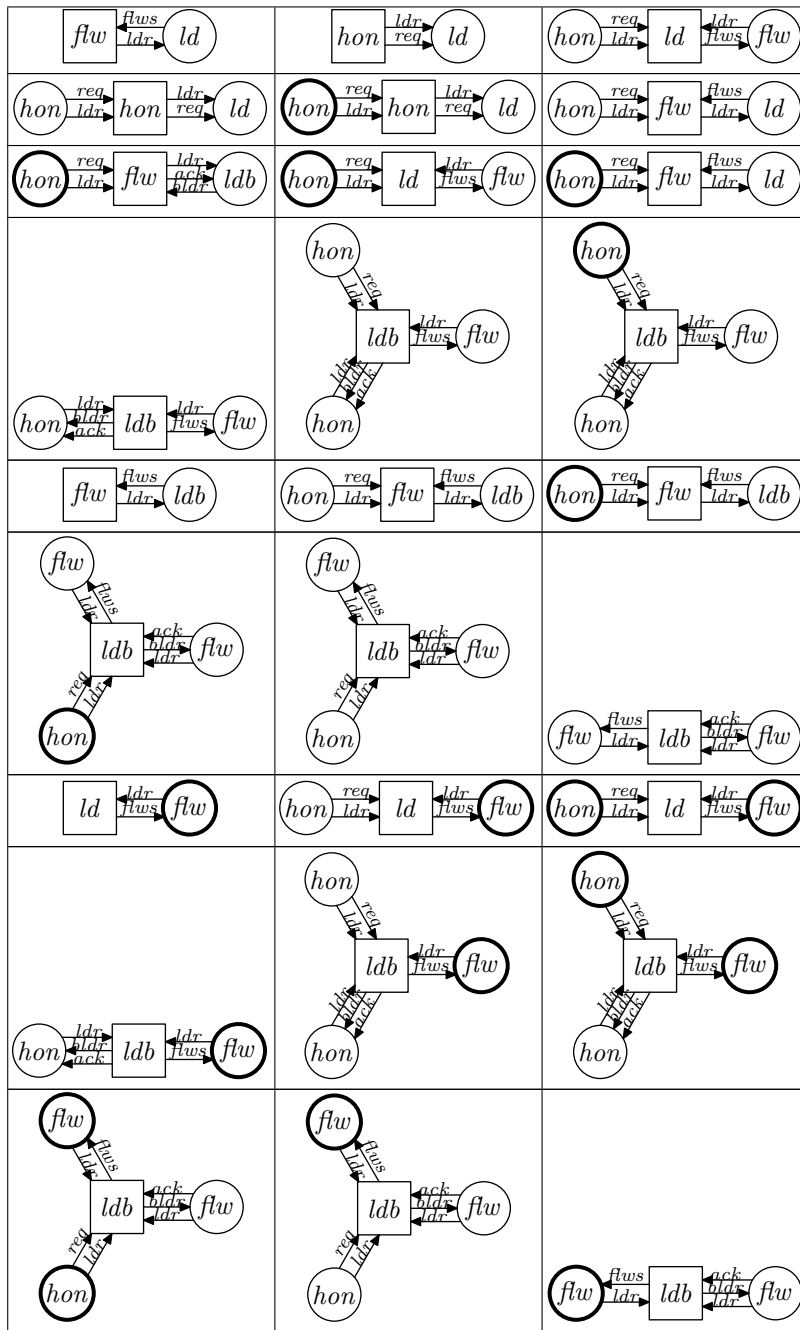
3 Results

Our tool, **astra**, expects a file in **hiralysis** format as input. The file contains a set of graph transformation rules and a start graph. We use the rules from the case study, except for the ones that deal with platoon merging and follower handover:



astra computes an abstraction of all graphs that can be generated by the system, resulting in the following set of stars:

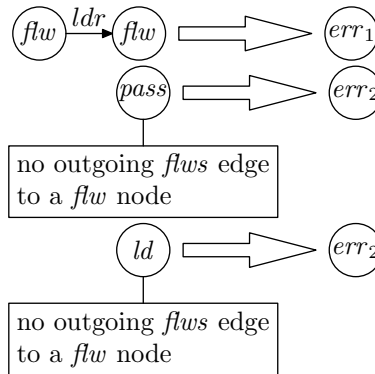
fa	$hon \xrightarrow[req]{ldr} fa$	$fa \xrightarrow[req]{ldr} hon$
$fa \xrightarrow[req]{ldr} hon$	$hon \xrightarrow[ldr]{req} hon \xrightarrow[req]{ldr} fa$	$hon \xrightarrow[req]{ldr} hon$
$hon \xrightarrow[ldr]{req} hon \xrightarrow[req]{ldr} hon$	$hon \xrightarrow[req]{ldr} hon$	$ldb \xrightarrow[ldr]{ack} hon$
$hon \xrightarrow[ack]{bldr} ldb$	$hon \xrightarrow[ldr]{req} hon \xrightarrow[req]{ldr} fa$	$hon \xrightarrow[req]{ldr} ldb$
$hon \xrightarrow[ack]{bldr} ldb \xrightarrow[req]{ldr} hon$	$hon \xrightarrow[ldr]{req} hon \xrightarrow[req]{ldr} ldb$	$hon \xrightarrow[req]{ldr} hon \xrightarrow[req]{ldr} ldb$
$hon \xrightarrow[ldr]{req} hon \xrightarrow[ack]{bldr} ldb$	$hon \xrightarrow[ldr]{req} hon \xrightarrow[ldr]{req} hon$	$hon \xrightarrow[ldr]{req} hon \xrightarrow[req]{ldr} hon$
$hon \xrightarrow[ldr]{req} hon \xrightarrow[ldr]{req} hon$	$hon \xrightarrow[ack]{bldr} ldb \xrightarrow[req]{ldr} hon$	$ldb \xrightarrow[ldr]{ack} flw$
$flw \xrightarrow[ack]{bldr} ldb$	$hon \xrightarrow[ldr]{req} ldb \xrightarrow[ldr]{ack} flw$	$hon \xrightarrow[req]{ldr} flw$
$hon \xrightarrow[ldr]{req} flw \xrightarrow[ack]{bldr} ldb$	$hon \xrightarrow[ldr]{req} hon \xrightarrow[req]{ldr} flw$	$hon \xrightarrow[req]{ldr} hon \xrightarrow[req]{ldr} flw$
$hon \xrightarrow[ldr]{req} hon \xrightarrow[ack]{bldr} ldb$	$hon \xrightarrow[ldr]{req} ldb \xrightarrow[ldr]{ack} flw$	$ld \xrightarrow[flws]{ldr} flw$



Our tool outputs the result in graphviz, GDL, XGDL, Tulip and METAPOST format and such that it can be rendered/displayed with any tool capable of processing these formats. The graphical representation above is the METAPOST output.

4 Property evaluation

Star abstraction easily allows for evaluation of properties involving two nodes and the connections among them, such as the first and second example given in section 5.2 of the case study. This is because the stars contain each adjacent node of each node and the the edges between them. That is the only thing that the first two example properties talk about. Using the additional rules



we mark parts of the graph with error labels where the properties are violated. It is then easy to scan the result for such labels. We can verify any property that can be expressed using such rules.

Our abstraction overapproximates the set of concrete topologies that might arise. Thus, if a property of the mentioned kind (one that can be expressed by a transformation rule adding an error node) is satisfied by all topologies represented by our abstract result, we know that it is satisfied by all reachable concrete topologies as well. However, if it is not satisfied in the abstract result, it might still be satisfied for the concrete result.

The computing time and memory consumption of our tool is negligible (< 1 MB, < 1 sec) on any reasonably modern machine.

5 Conclusion

Our analysis has proven powerful enough to analyze the join phase of the merge protocol in a general way, without limit to the number of processes. The resulting topologies are more complex than what can be analyzed with existing approaches, since they are not limited with respect to path length.

On the other hand, the abstraction we employed is too weak to deal with the characteristic topology structures occurring during handover. If we try to

analyze the full merge protocol, the abstraction runs into state space explosion. This is caused by the inability of the abstraction to preserve the fact that the topology has a triangular shape during handover. Accordingly, the abstraction does not preserve enough information to verify the third example property from the case study.

Since the results are still promising, we are currently implementing a new tool with an extended abstraction that is able to cope with topologies involving triangular shapes.

References

1. Peter Backes. Topology analysis of dynamic communication systems. Diploma thesis, Universität des Saarlandes, March 2008.
2. Jörg Bauer. *Analysis of Communication Topologies by Partner Abstraction*. PhD thesis, Universität des Saarlandes, 2006.
3. Jörg Bauer and Reinhard Wilhelm. Static analysis of dynamic communication systems. In *14th International Static Analysis Symposium*, 2007.