Synthesizing Abstract Transformers for Reduced-Product Domains

PANKAJ KUMAR KALITA, Indian Institute of Technology Kanpur, India
THOMAS REPS, University of Wisconsin–Madison, USA
SUBHAJIT ROY, Indian Institute of Technology Kanpur, India

Contents

1 GETTING STARTED

This section provides instructions for setting up the artifact and running the evaluation. The artifact is packaged as a docker image and Section 1.1 provides steps to setting up docker-engine in your system. Your system should meet the following minimum requirements for artifact evaluation.

- OS : Ubuntu 18.04.
- RAM : 32GB or higher is recommended.
- CPU (cores): CPU with eight or more logical cores. (Intel i7 8th Gen or higher.)
- Secondary Memory: We recommend at least 20 GB of free space for all the experimental evaluations to be conducted.
- Internet Access: We require a working internet connection to download the artifact.

1.1 Starting with the Docker

Use the following instructions to install Docker in Ubuntu. One can skip the following instructions to install docker if it is already installed.

```
$ sudo apt-get install docker.io
```

One can verify whether the docker has been installed or not.

```
$ sudo docker run hello-world
```

1.1.1 Running Docker from Zenodo. Download the tar file containing the docker image from Zenodo. Now open the terminal from the downloaded folder and type the following command to load the image.

```
$ sudo docker load < amurth2_SAS2024.tar.gz
```

This will load the docker image. Use the following code to run the image.

```
$ mkdir genResult
$ sudo docker run -it -v $(pwd)/genResult:/root/genResult amurth2:V1
```

1.1.2 Docker pull from the Docker Hub. Please pull our docker image from the docker hub using the following code snippet.

```
$ sudo docker pull pkalita595/amurth2:V1
```

After completion of the docker download, we are ready to start the docker using the following command.

```
$ mkdir genResult
$ sudo docker run -it -v $(pwd)/genResult:/root/genResult pkalita595/amurth2:V1
```

Directory `genResult` can be used to store relevant logs/plots to be accessed from outside the container. After starting the docker, please change the working directory to `/root` using the following.

```
$ cd /root
```

1.2 Artifact structure

In `/root` directory of the artifact, there are two directories as shown below.

```
$ ls
 amurth2 genResult
```

2 STEP-BY-STEP INSTRUCTIONS

2.1 Experiment for transformer synthesis using Amurth2

2.1.1 Running a simple example. In this section, we will see how to run a single benchmark using Amurth2. Please change the directory to `amurth2` .

```
$ cd /root/amurth2
```

In `config` directory we put `json` files for all the operations we have synthesized. For each domain and each operation, we have provided a `json` file. All of our source code is present in `src` directory. Please change the current directory to `src` .

```
$ cd src
```

We have provided the Python script `run.py` to run a single benchmark from the set of benchmarks present in `config` directory. This script takes two arguments, i.e., `<domain name>` and `<operation name>` . If we want to synthesize a transformer for `addition` (`add`) operator in the Odd interval and Even interval domain, you will type as follows.

```
$ python run.py oe add
```

This will run `amurth2.py` with the `json` file related to the `abs` operator for Odd interval and Even interval domain (`oe`). This will run for some time and in the end, will produce the transformer.

Following is the list of domains and operators Amurth2 supports along with the IDs to pass to the `run.py` .

Domain name	SAFE	JSAI	Odd-Even Interval
ID	safe	jsai	oe

Following is the list of IDs of all the operations.

Arithmetic	add	sub	abs (absolute)	inc (increment)		
String	concat	charAt	contains	toLower	toUpper	trim

Amurth2 generates some temporary files while running. Sketch files generated during each iteration of precision check, MaxSAT synthesize can be found in `amurth2/temp/sketch` . If reviewer runs experiment using `run.py` , then complete log of the operation can be found in `amurth2/temp/` directory. Naming convention of the log will be like `log_<operation name>.txt`. Here `<operation name>` is the name of operation present in the `tosynthesize` field in the respective `json` file. *In case of failure please run again.*

2.1.2 Understanding input and output of Amurth2. Please refer to README of Amurth artifact [1].

2.2 Structure of the Amurth2

Directory `/root/amurth2` contains all the source code and benchmarks for synthesizing abstract transformers. Following is the detailed breakdown of the tool directory structure.

- `abstract_domain` : Contains files that defines the domains in sketch and files for bootstrapping examples.
- `aux_function` : Contains auxiliary functions required during synthesis.
- `config` : It contains `json` files for each operation in different domains. In case Amurth2 we describe each input to the tool with the help of `json` files. One can assume that each `json` file is one benchmark. `json` files for each domains are organised in the directories inside `config` .
- `dsl` : It contains DSL (domain-specific language) for each domain in separate directories.
- `external_lib` : Contains definitions of the concrete operators.
- `include` : Include files for sketch to run.
- `src` : This contains all the source code for Amurth2. `amurth2.py` is the main driver file to run Amurth2. `lib` directory contains library functions source code.
- `temp` : Contains temporary files created during the running of the tool.

REFERENCES

[1] Pankaj Kumar Kalita, Sujit Muduli, Loris D'Antoni, Thomas Reps, and Subhajit Roy. 2022. Synthesizing Abstract Transformers. https://doi.org/10.5281/zenodo.7092952