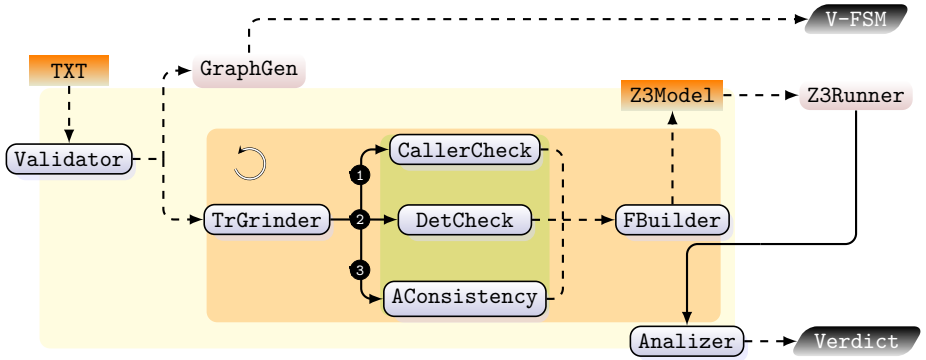
Verification

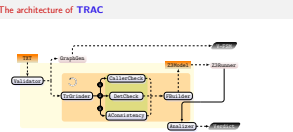
Checking well-formedness by hand is laborious and cumbersome

So we implemented **TRAC**, which
- transforms DAFSMs in a DSL to specify DAFSMs
- verifies well-formedness condition relying on the SMT solver Z3

Verification
Checking well-formedness by hand is laborious and cumbersome
So we implemented TRAC, which
transforms DAFSMs in a DSL to specify DAFSMs
verifies well-formedness condition relying on the SMT solver Z3

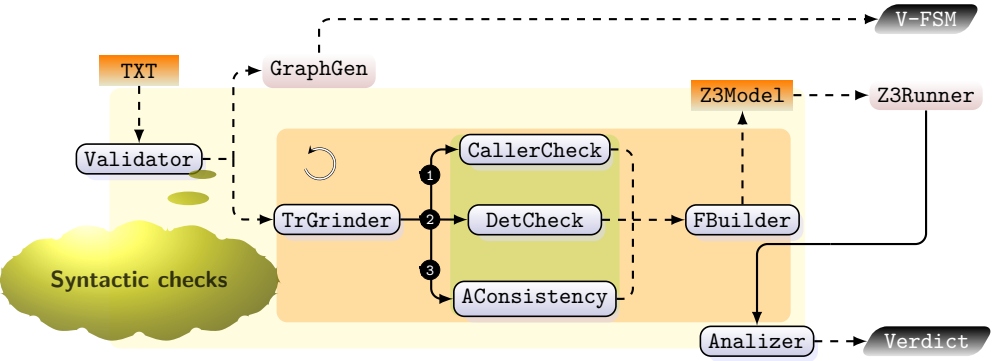
The architecture of **TRAC**

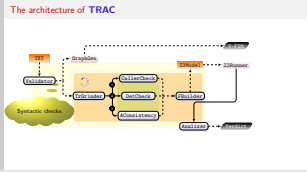




the architecture of **TRAC** is compartmentalised into two principal modules:
parsing and visualisation (yellow box) and
TRAC's core (orange box). The latter module implements well-formedness check (green box).
Solid arrows represent calls between components while dashed arrows data IO.

The architecture of **TRAC**

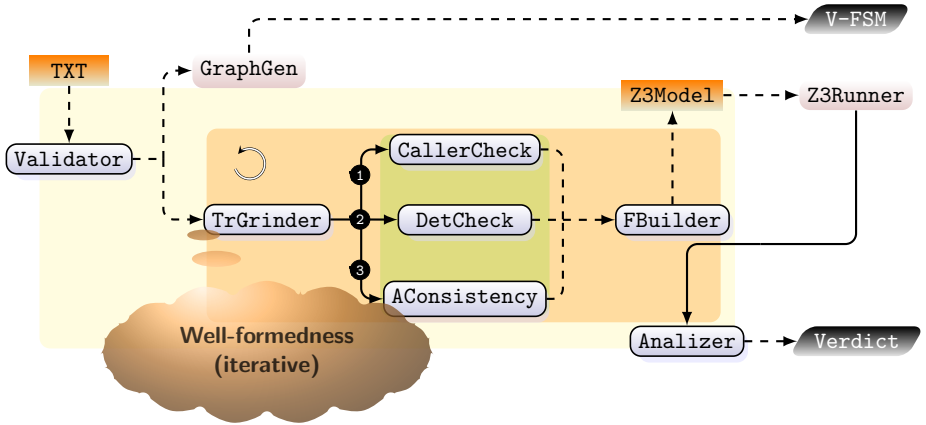


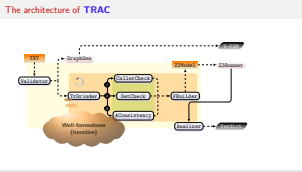


basic syntactic checks on a DSL representation of DAFSMs and transforming the input in a format that simplifies the analysis of the following phases:

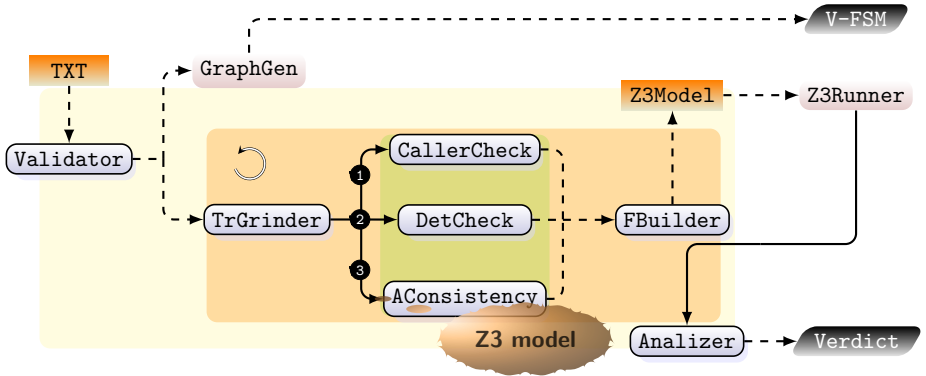
- passed to `GraphGen` for visual representation of DAFSMs (`V-FSM` output)
- passed to the `TrGrinder` component (orange box) for well-formedness checking.

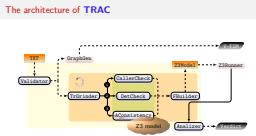
The architecture of **TRAC**





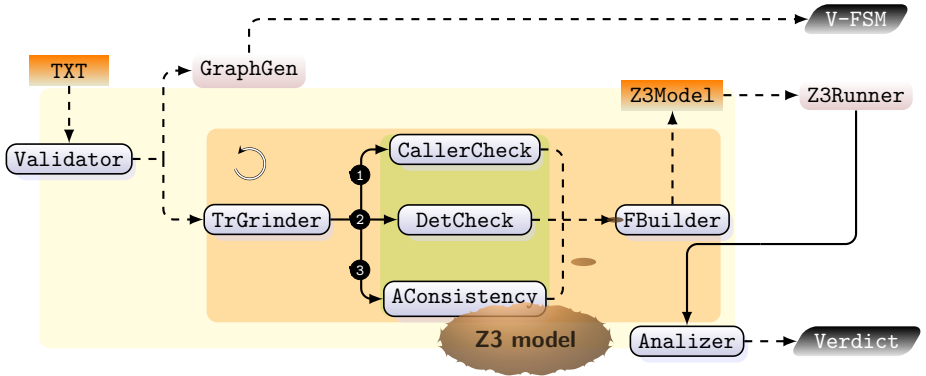
The architecture of **TRAC**

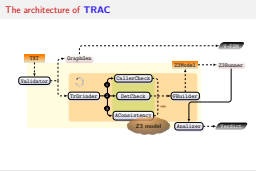




`AConsistency` (arrow ❸) to generate a Z3 formula which holds if, and only if, the transtion is consistent.

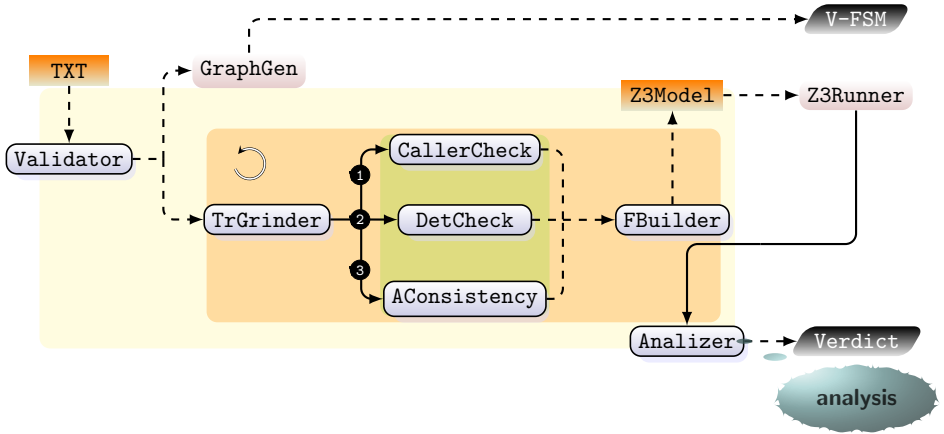
The architecture of **TRAC**

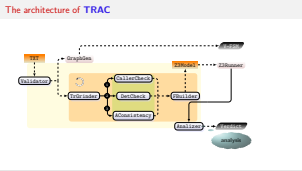




computes the z3 f.la equivalent to the conjunction of the outputs which is then passed to a Z3 engine to check its satisfiability

The architecture of **TRAC**





Finally, the `Analizer` component that diagnoses the output of Z3 and produces a `Verdict` which reports (if any) the violations of well-formedness of the DAFSM in input.

Installation

Detailed instructions at `https://github.com/loctet/TRAC`

Dependencies: Java RE (to render DAFSM graphically) & Python 3.6 or later

```
pip install z3-solver matplotlib numpy plotly pandas networkx
```


forse servono solo solo z3-solver e networkx
https://doi.org/10.5281/zenodo.10996456