T2: TEMPORAL PROPERTY VERIFICATION

Heidy Khlaaf[1]

Marc Brockschmidt[2] Byron Cook[1] Samin Ishtiaq[2] Nir Piterman[3]

University College London[1]

Microsoft Research[2]

University of Leicester[3]

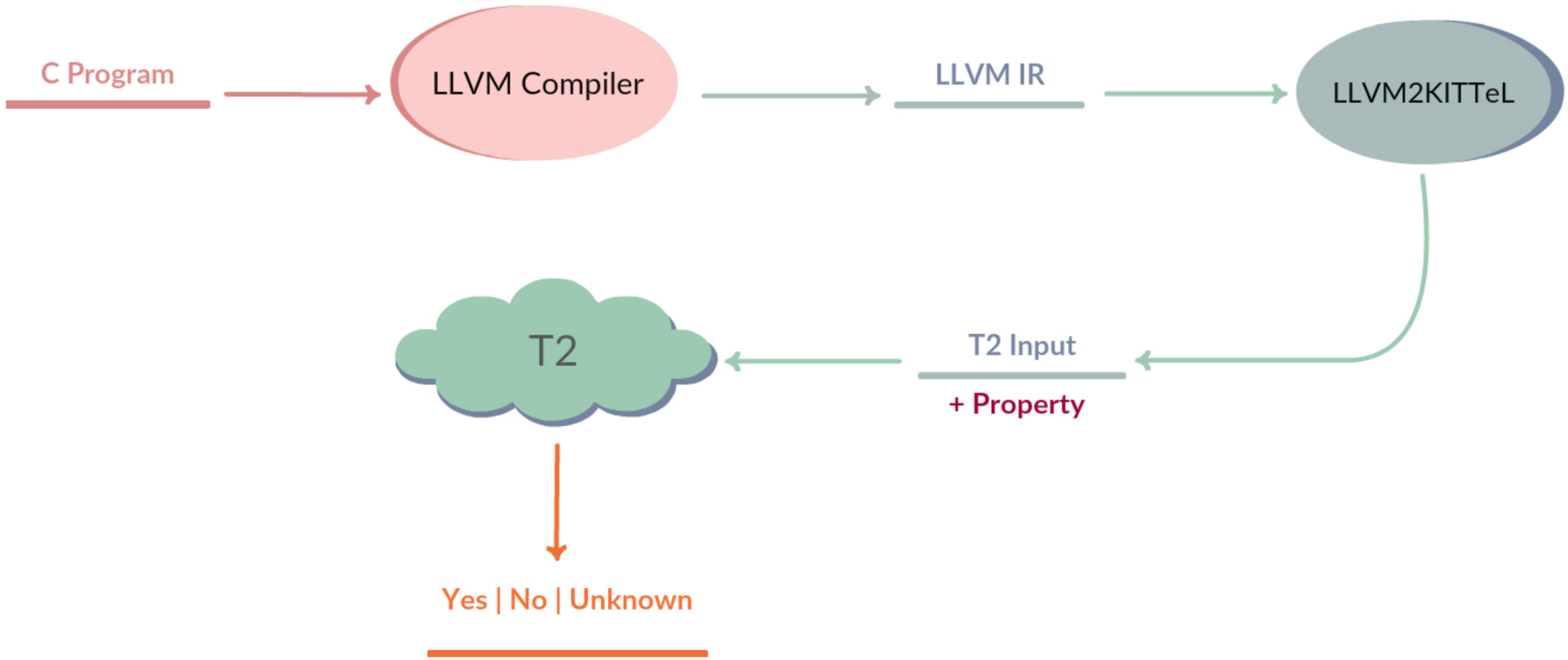
OVERVIEW

- First open-source, public release of **T2** (TERMINATOR 2), a follow-up of the TERMINATOR project.

- Supports automatic verification of temporal-logics (**CTL, Fair-CTL, CTL***) and user-provided liveness and safety properties over (integer) infinite-state systems.

- Input can be provided directly in **C** or other languages via support of the **LLVM** compiler framework.

OVERVIEW


C Program
LLVM Compiler
LLVM IR
LLVM2KITTeL
T2
T2 Input
+ Property
Yes | No | Unknown

T2: FEATURES

- Verification of temporal logic (reasoning about propositions qualified in terms of time).

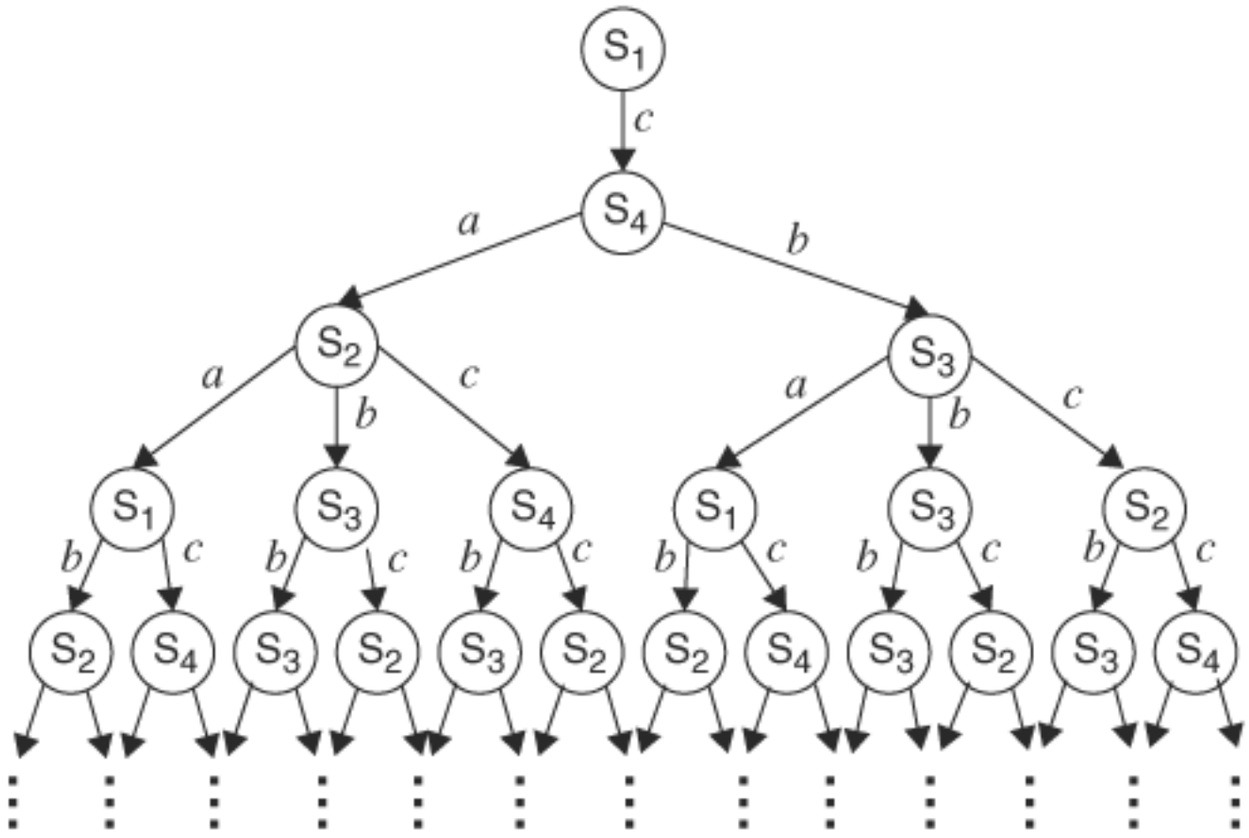
- Encompasses safety, termination, liveness, fairness, etc.

- Supported sub-logics are: **CTL, Fair-CTL, CTL*.**

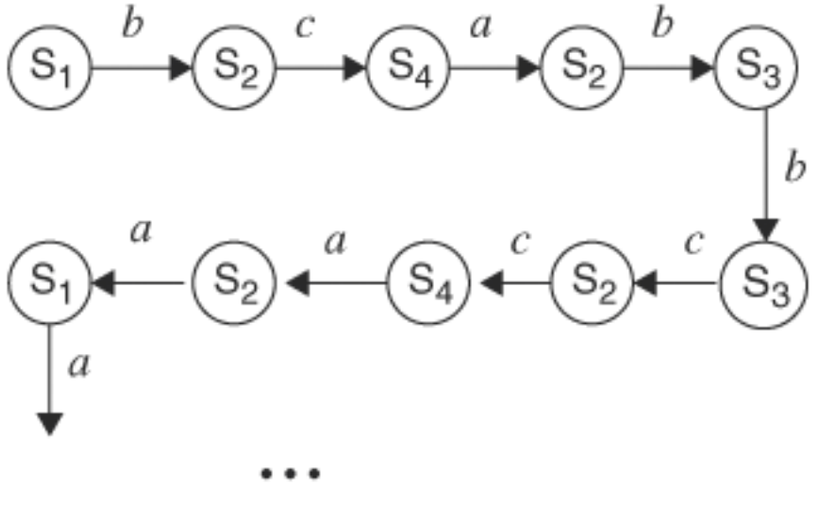
T2: FEATURES

- T2 is the only tool that can handle automated verification of **Fair-CTL and CTL*** for infinite-state(integer) systems.
- QARMC/HSF handles **CTL** but requires horn clause constraints to be provided by the user as input.

CTL*

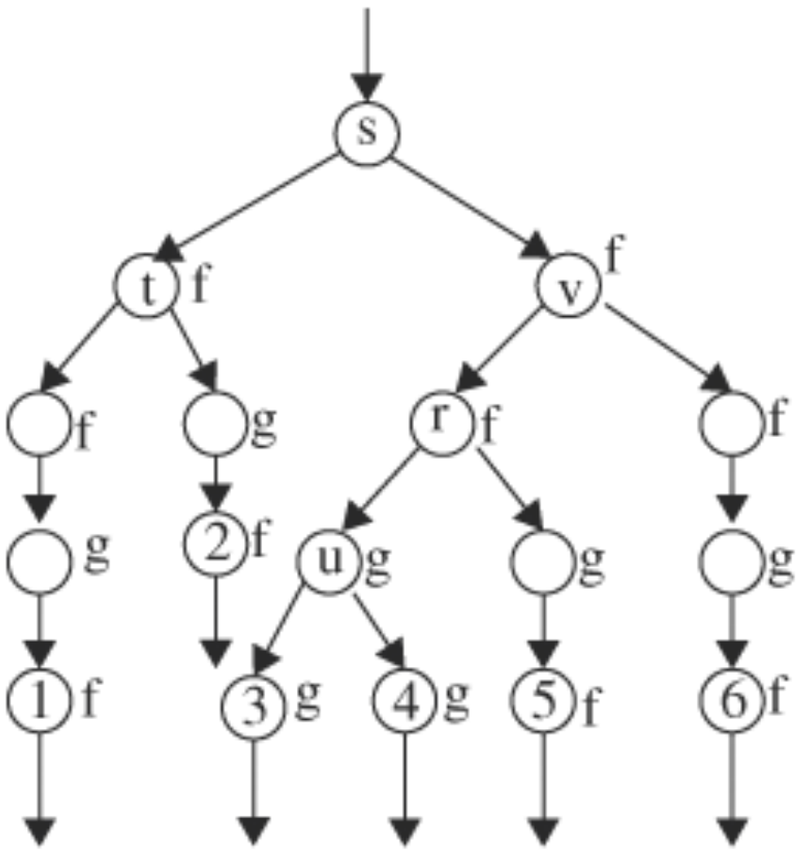

S1
c
S4
a
b
S2
S3
a
c
b
a
b
c
S1
S3
S4
S1
S3
S2
b
c
b
c
b
c
b
c
b
c
b
c
S2
S4
S3
S2
S3
S2
S2
S4
S3
S2
S3
S4

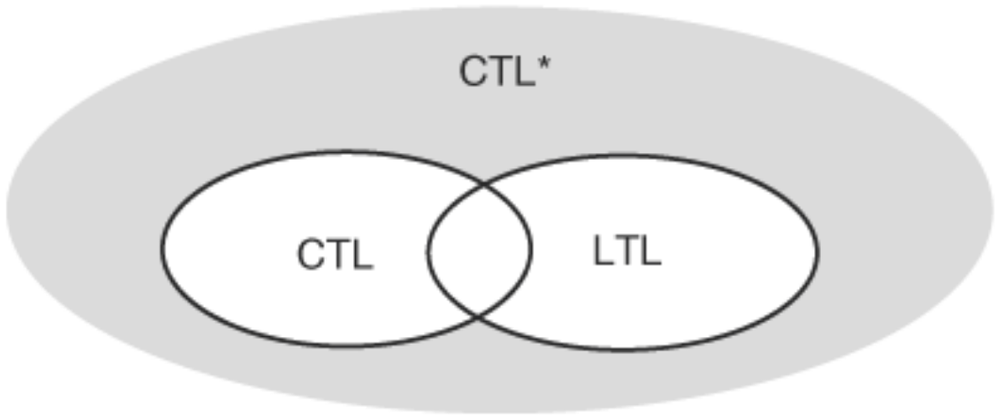
CTL


S1
b
S2
c
S4
a
S2
b
S3
b
S1
a
S2
a
S4
c
S2
c
S3
a
...

LTL

CTL*





T2: FEATURES

- Eventually this session will end.
  - AF (Session ends).
- There exists a sequence of actions that infinitely often leads to the coffee break table with pastries.
  - EGF (Coffee Table && Pastries)

T2: BACK-END

- "*Fairness for Infinite-State Systems*", TACAS'15
  "*On Automation of CTL* Verification for Infinite-State Systems*", CAV'15

  - Reduce the verification of CTL* and Fair-CTL to a CTL Model-Checking problem.

    - Via prophecy variables and program instrumentation.

- "*Faster Temporal Reasoning for Infinite-State Programs*", FMCAD'14

  - CTL can be reduced to a termination and safety problem via program instrumentation.

T2: BACK-END

CTL*/Fair-CTL **CTL** **Termination** **Safety**

T2: BACK-END

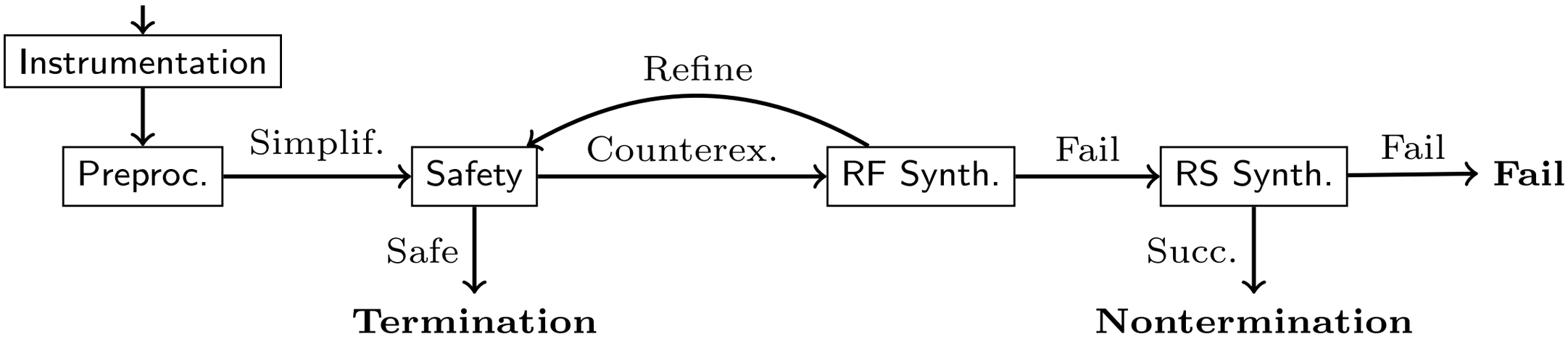
- Builds upon safety proving procedures: **Impact**, **Z3**, and **Spacer.**

- **Termination** back-end constructs a termination proof through a sequence of safety queries and ranking function synthesis steps.

  - *"Ramsey vs. lexicographic termination proving"*, TACAS'13

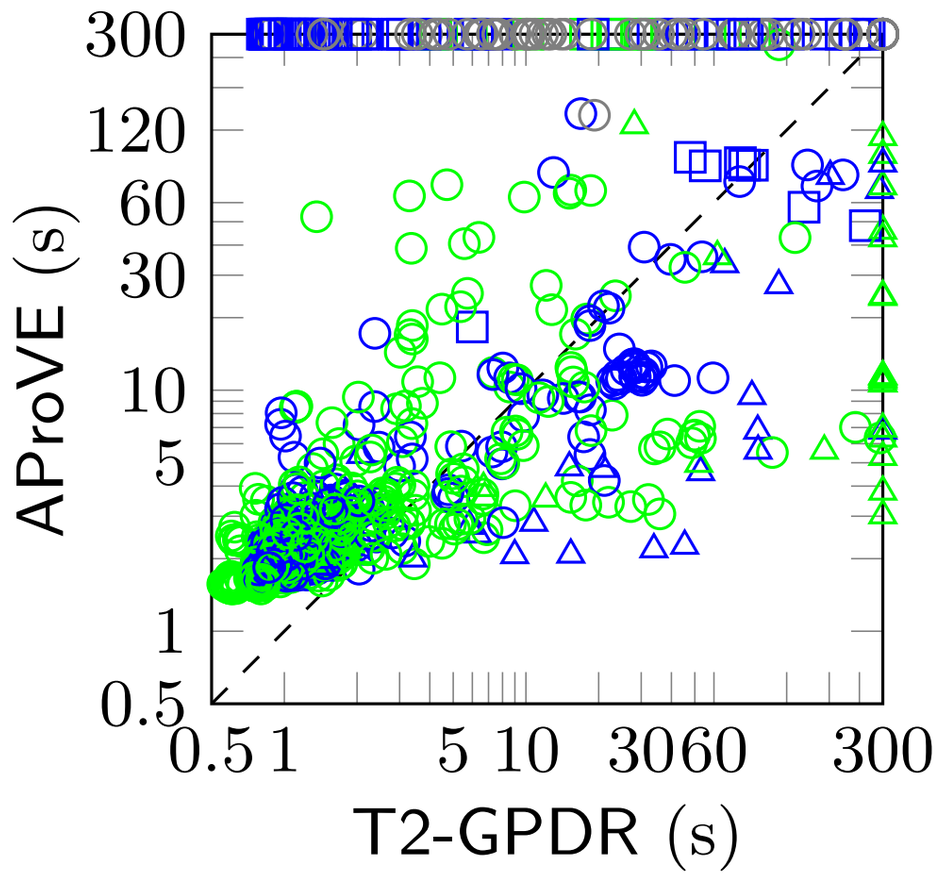
  - *"Better termination proving through cooperation"*, CAV'13

T2: BACK-END


Instrumentation
Preproc.
Simplif.
Safety
Refine
Counterex.
RF Synth.
Fail
RS Synth.
Fail
Fail
Safe
Termination
Succ.
Nontermination

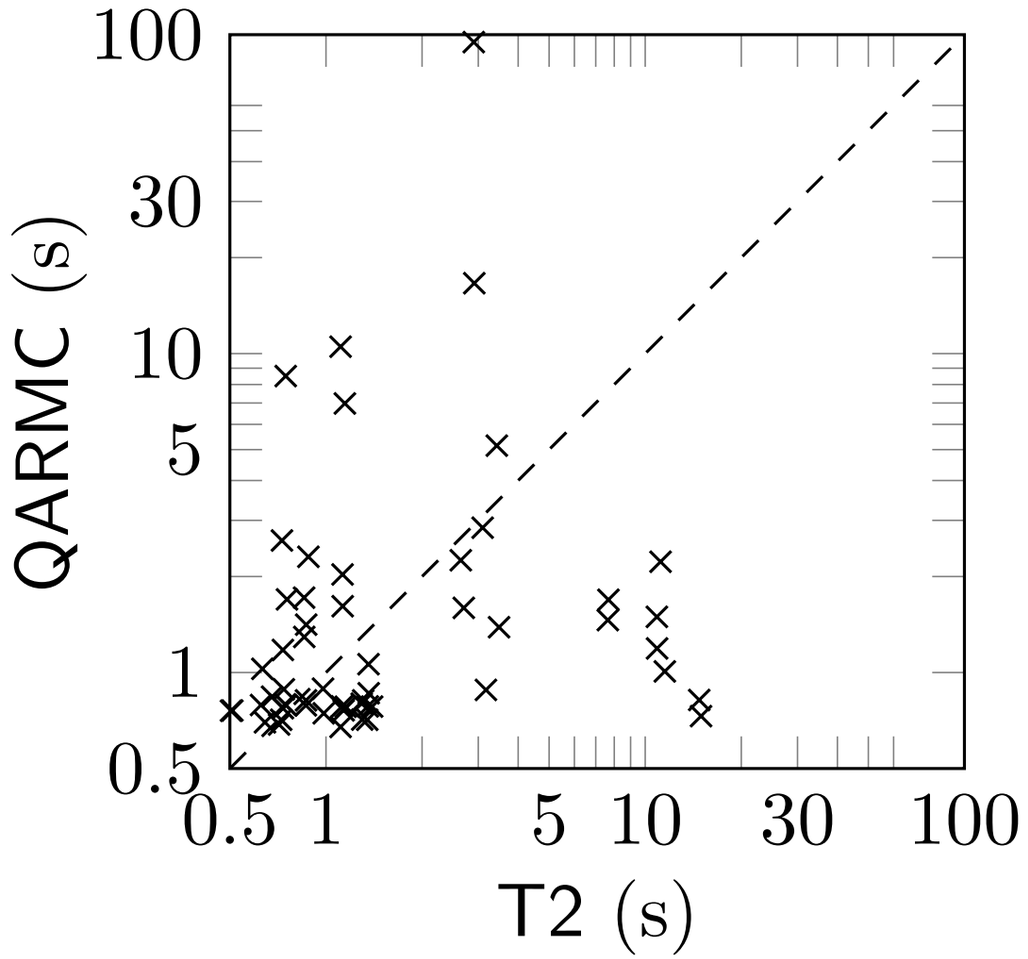
EXPERIMENTS: TERMINATION

Tool	**Term**	**Nonterm**	**Fail**	**Avg.** (s)
AProVE	641	393	188	49.1
CppInv	566	374	282	65.5
Ctrl	445	0	777	80.0
T2-GPDR	627	442	153	23.6
T2-GPDR-NoP	589	438	195	31.4
T2-Spacer-NoP	591	429	202	33.5
T2-Impact-NoP	529	452	241	37.2



- 1222 termination proving benchmarks from Termination Competition 2015.

EXPERIMENTS: CTL



- 56 benchmarks where T2 takes 2.7 seconds on average and Q'ARMC takes 3.6 seconds.

T2: RECAP

- Supports automatic verification of **CTL, Fair-CTL, CTL*, termination** and **safety** properties over (integer) infinite-state systems.

- Open-Source: https://github.com/mmjb/T2

- Supports **LLVM** languages via LLVM2KITTeL +T2 extension: https://github.com/hkhlaaf/llvm2kittel

- For a close-up **demo**: TACAS Tool Market
  Room: Outside Blauwe Zaal, floor 1