CVC4 for Sygus Comp 2018

- CVC4 is an SMT solver
  - Fourth generation of Cooperating Validity Checker (CVC, CVC Lite, CVC3, CVC4)
  - Supports many theories:
    - Linear arithmetic, bitvectors, UF, datatypes, arrays, sets, *strings, floating points* …
  - Two approaches for refutation-based synthesis in SMT **[Reynolds et al CAV 15]**
    1. Counterexample-Guided $\forall$ Instantiation for single invocation properties
    2. Techniques Enumerative syntax-guided synthesis

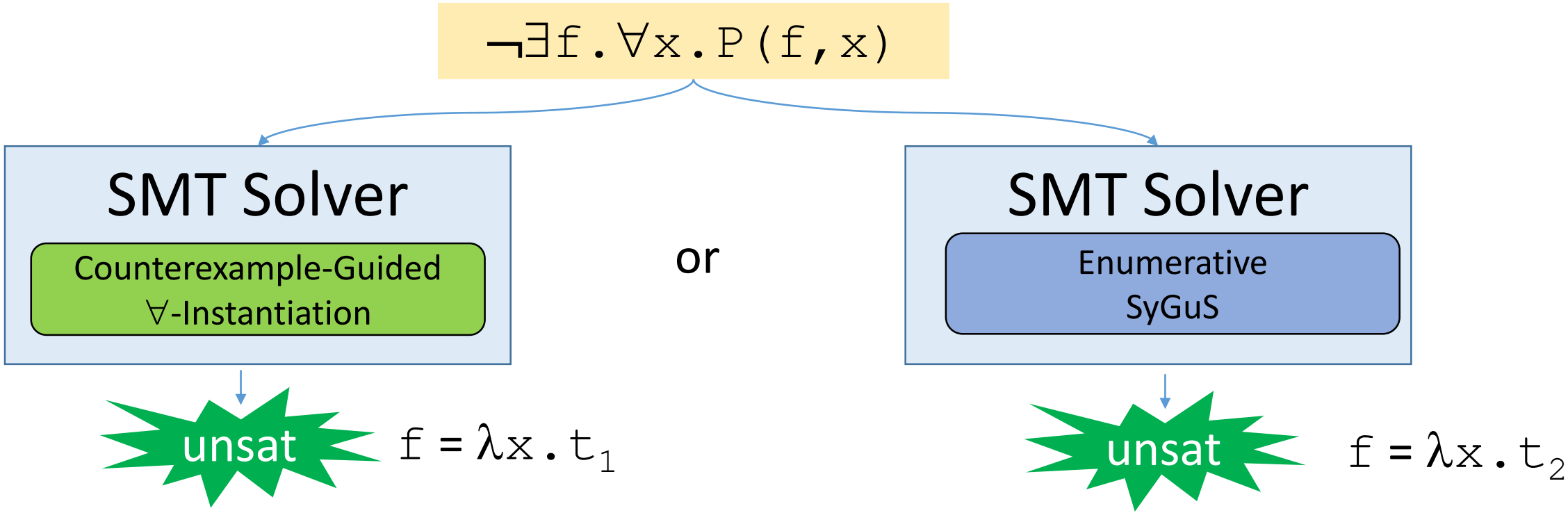
    $\Rightarrow$ …and some approaches that combine the two

Refutation-Based Synthesis in SMT

$$\neg\exists f.\forall x.P(f,x) \quad + \quad [\ \mathcal{R}\ ]$$

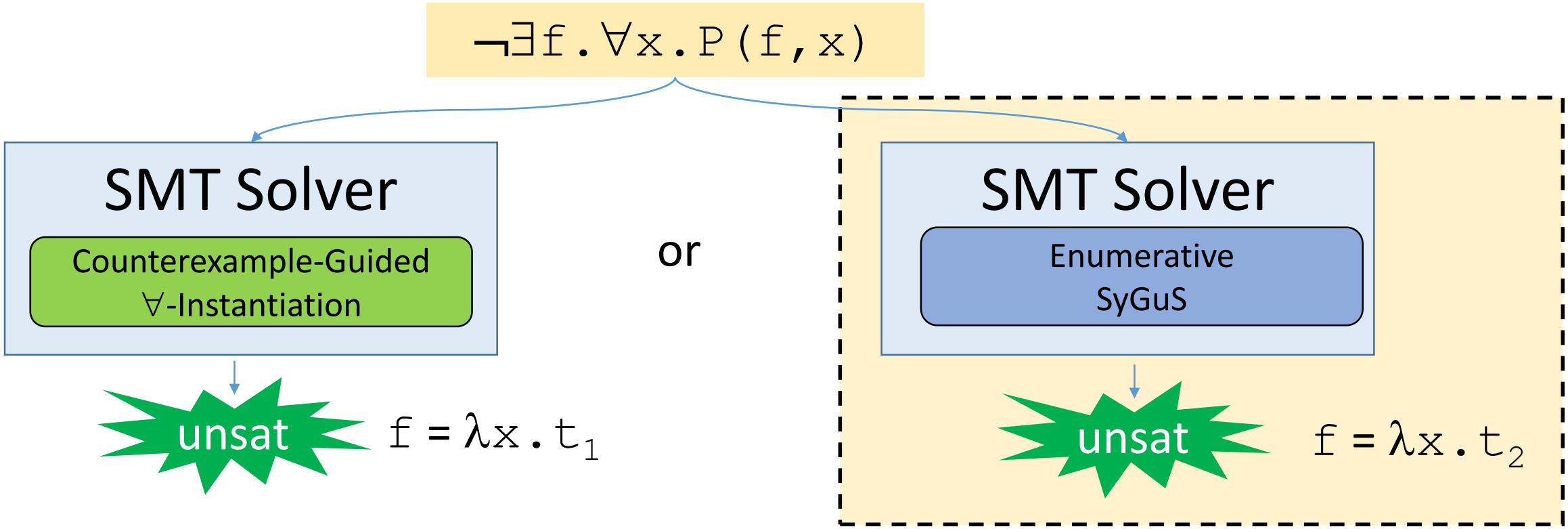
Negated Synthesis Conjecture
(+ syntactic restrictions $\mathcal{R}$)

Refutation-Based Synthesis in SMT



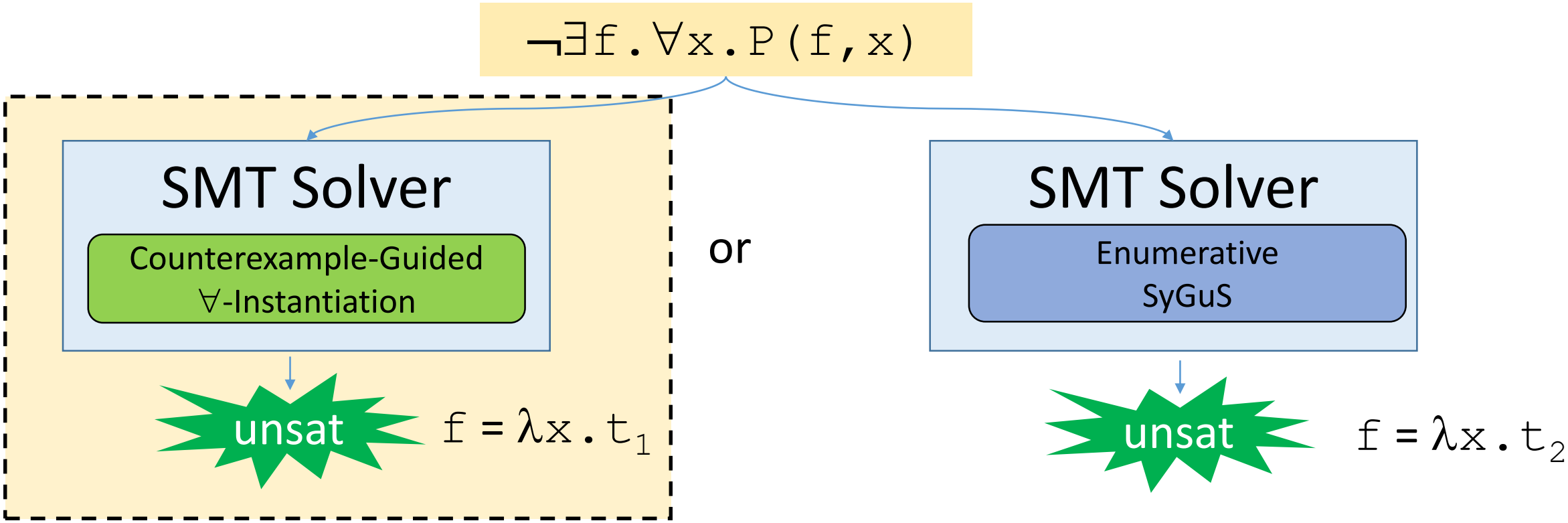
- Two approaches for refutation-based synthesis in SMT solvers **[Reynolds et al CAV2015]**

Refutation-Based Synthesis in SMT



⇒ Based on enumerative search (via syntax-guided synthesis) **[Alur et al 2013]**

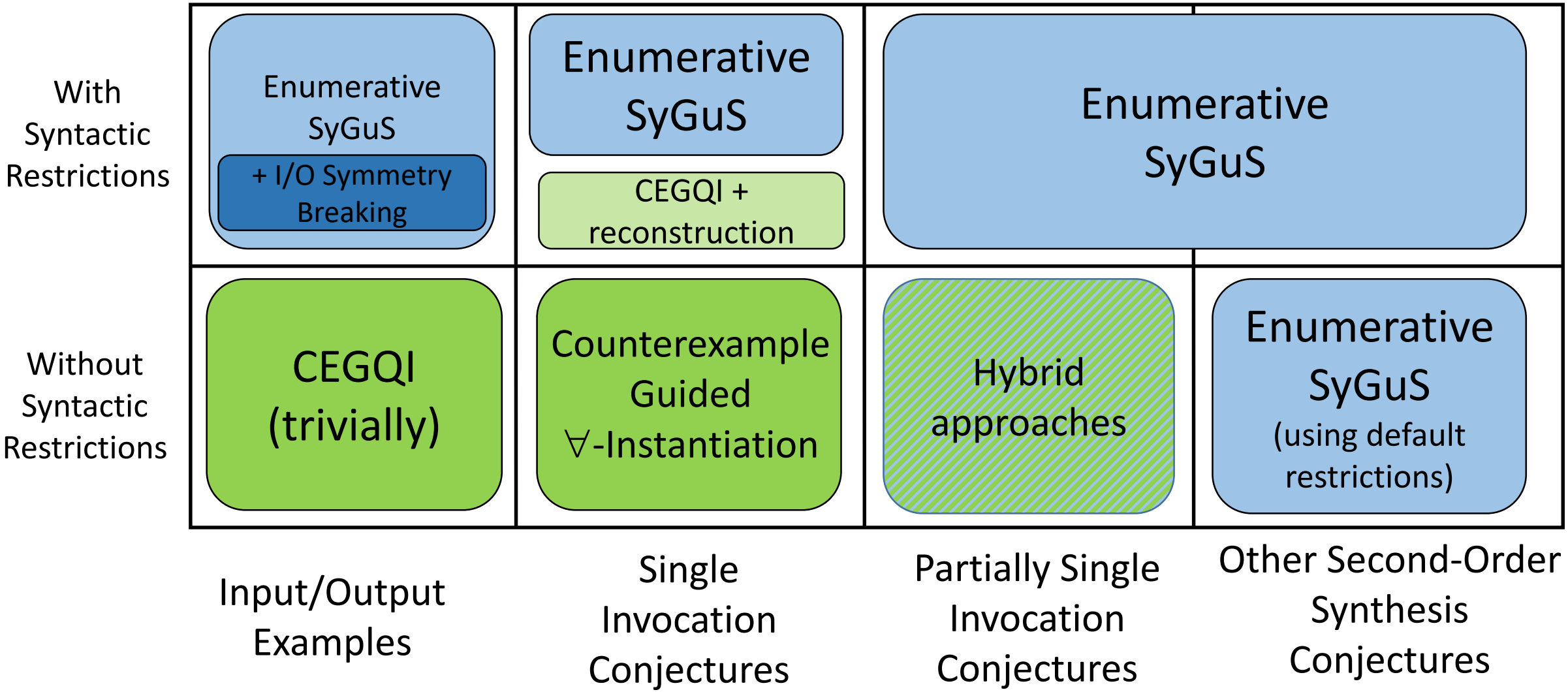
Refutation-Based Synthesis in SMT



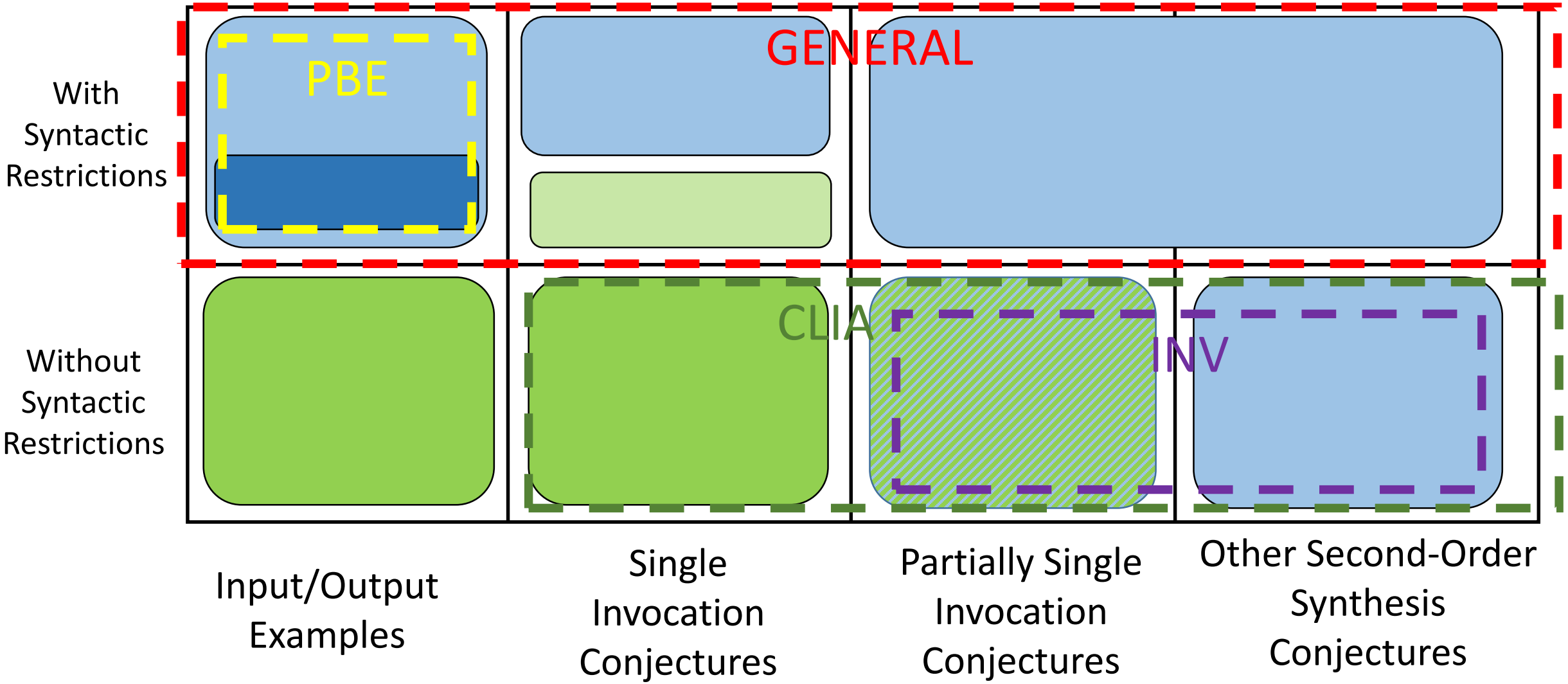
⇒ Based on first-order quantifier instantiation in SMT

[Monniaux 2010, Bjorner 2012, Komuravelli et al 2014, Dutertre 2015…]

CVC4 for Sygus Comp 2018



CVC4 for Sygus Comp 2018



Key Optimizations

- For enumerative SyGuS approach (**all tracks**):
  - Use of “shared selector” datatypes to reduce #terms **[Reynolds et al IJCAR 2018]**
    - Efficient encoding of enumerative search into SAT
  - Aggressive rewriting techniques for strings, BV, Booleans **[Reynolds et al SMT 2018]**
    - Avoid T-equivalent terms using SMT solver’s rewriting techniques as oracle
  - Constant Repair **[Abate et al CAV2018]**
    - Use of explicit “any constant” constructor which may be filled by subcalls to CVC4’s $\exists\forall$ solver
  - “Conjecture-specific symmetry breaking” (ongoing)
    - Use aspects of the conjecture to statically limit the set of possible solutions
- For **INV and PBE tracks**:
  - Implementation of divide-and-conquer techniques, inspired by **[Alur et al TACAS2017]**
    - Decision tree learning for `ite`-solutions (including predicates)
    - “Successive prefixes” algorithm for `concat`-solutions for PBE Strings