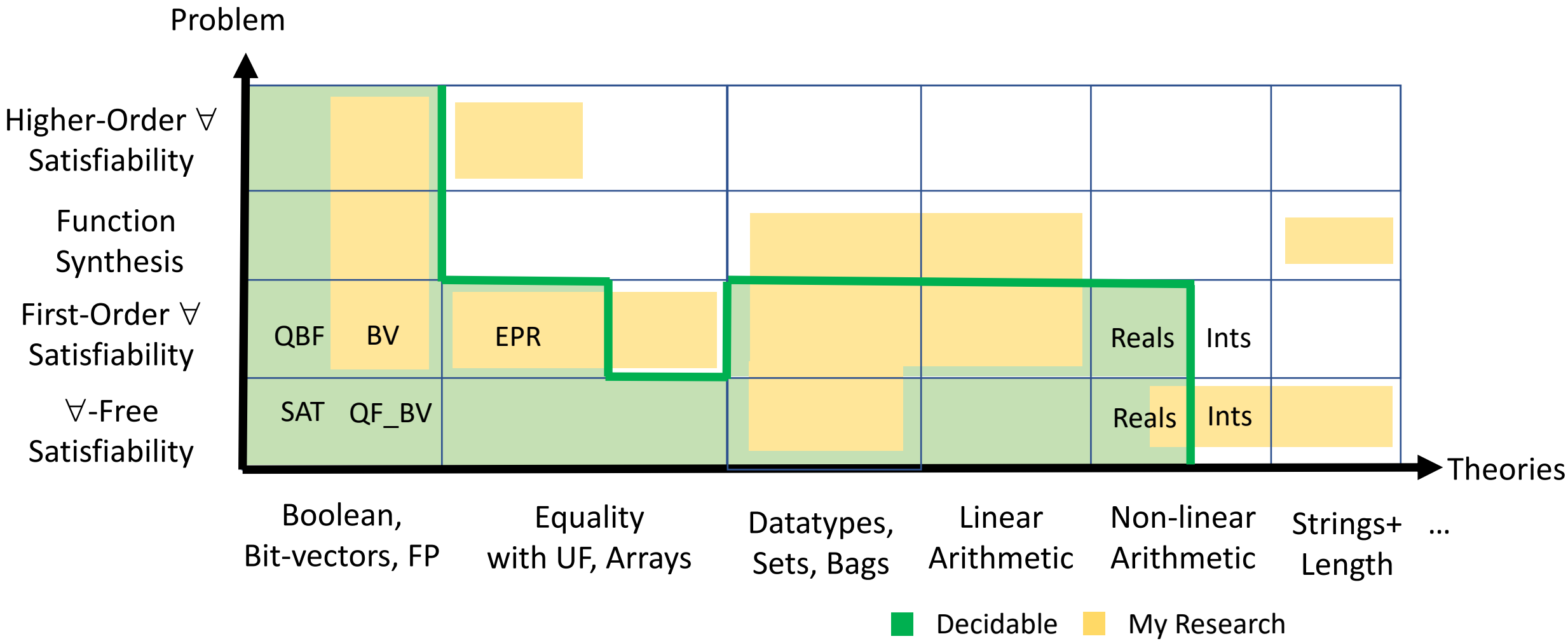
…some of the Secrets of cvc5

Andrew Reynolds

Centaur Annual Meeting 2022

July 12, 2022

cvc5: SMT and beyond

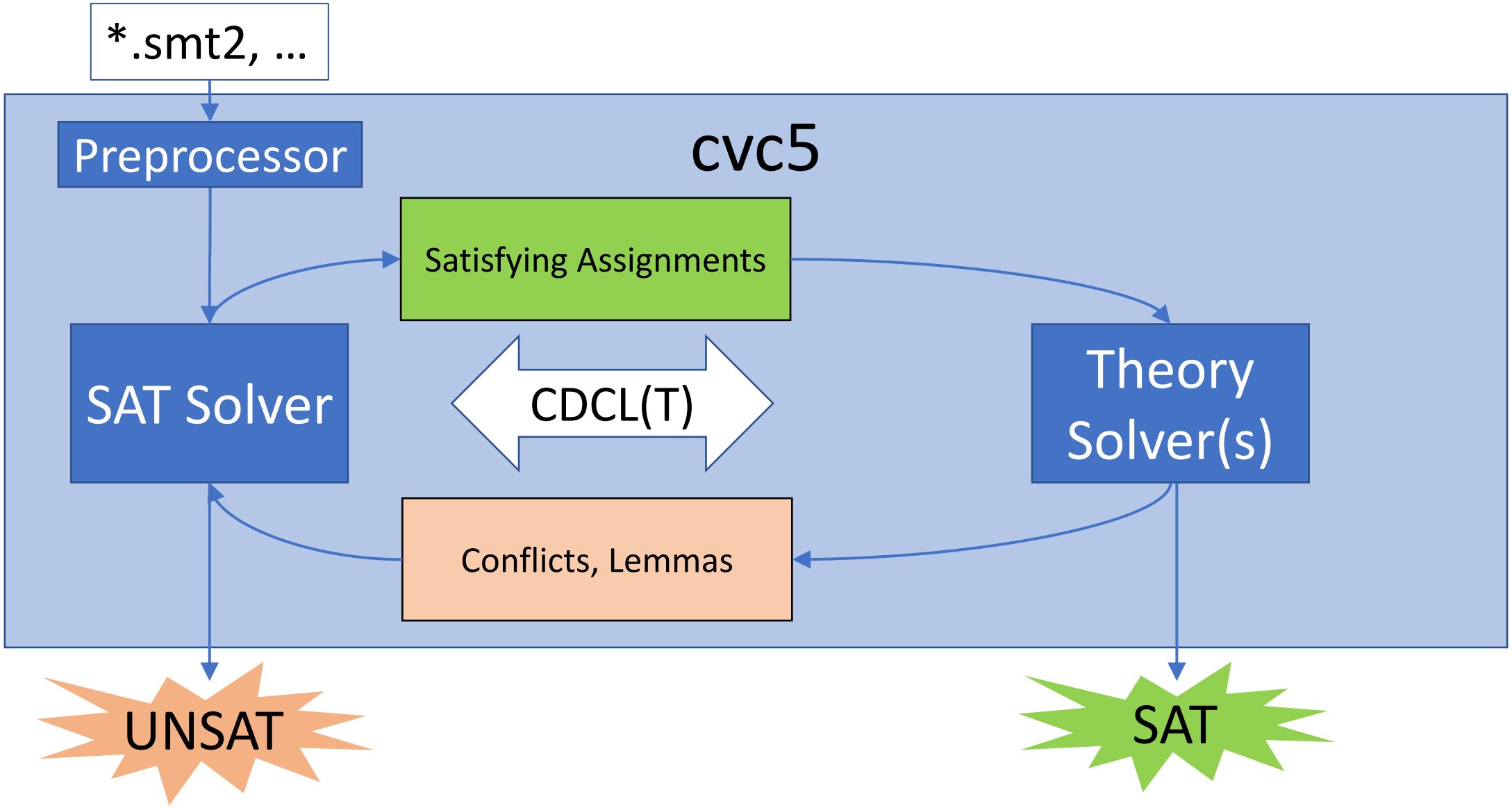


cvc5: a Versatile and Industrial-Strength SMT Solver

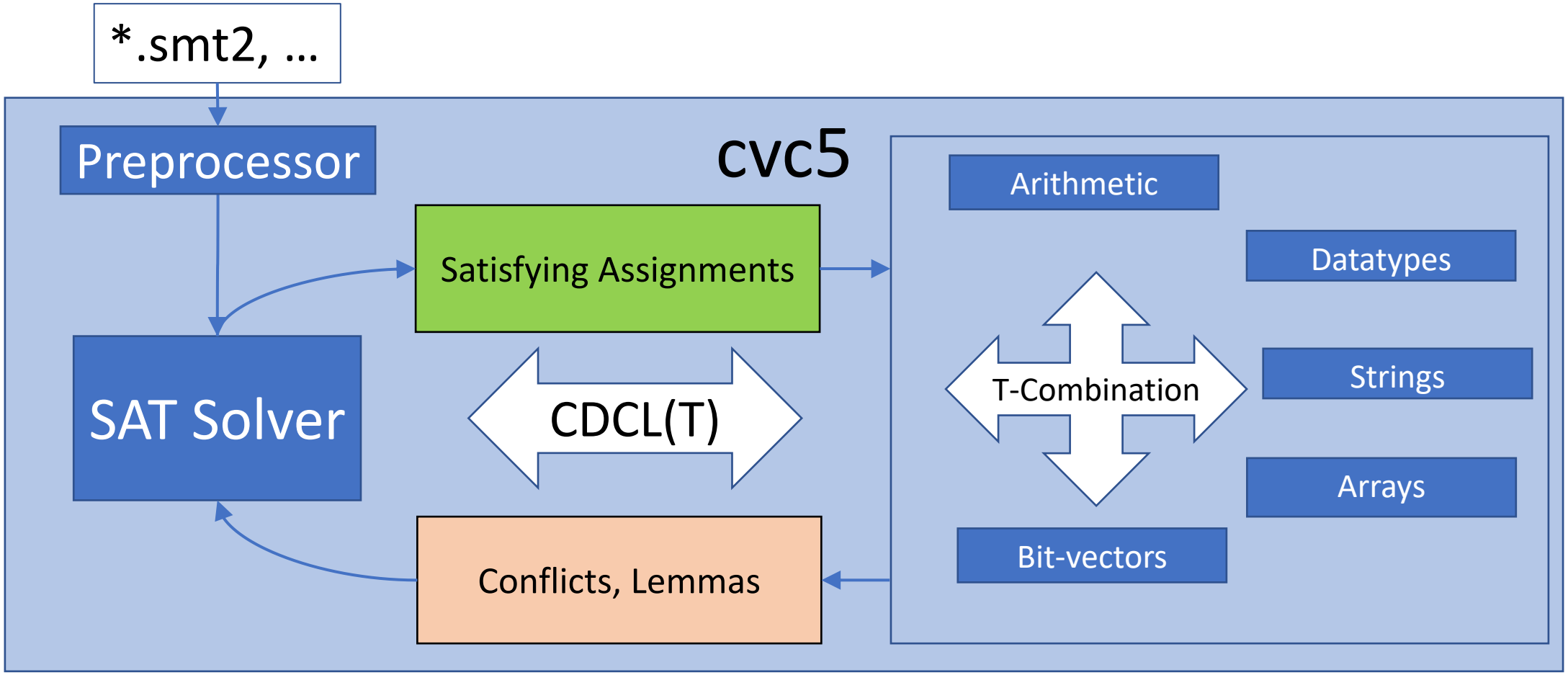
- Architecture and the design of theory solvers
  - Focus on theory of strings

- Future Directions in cvc5
  - Deep restarts
  - Difficulty estimates

Architecture of cvc5

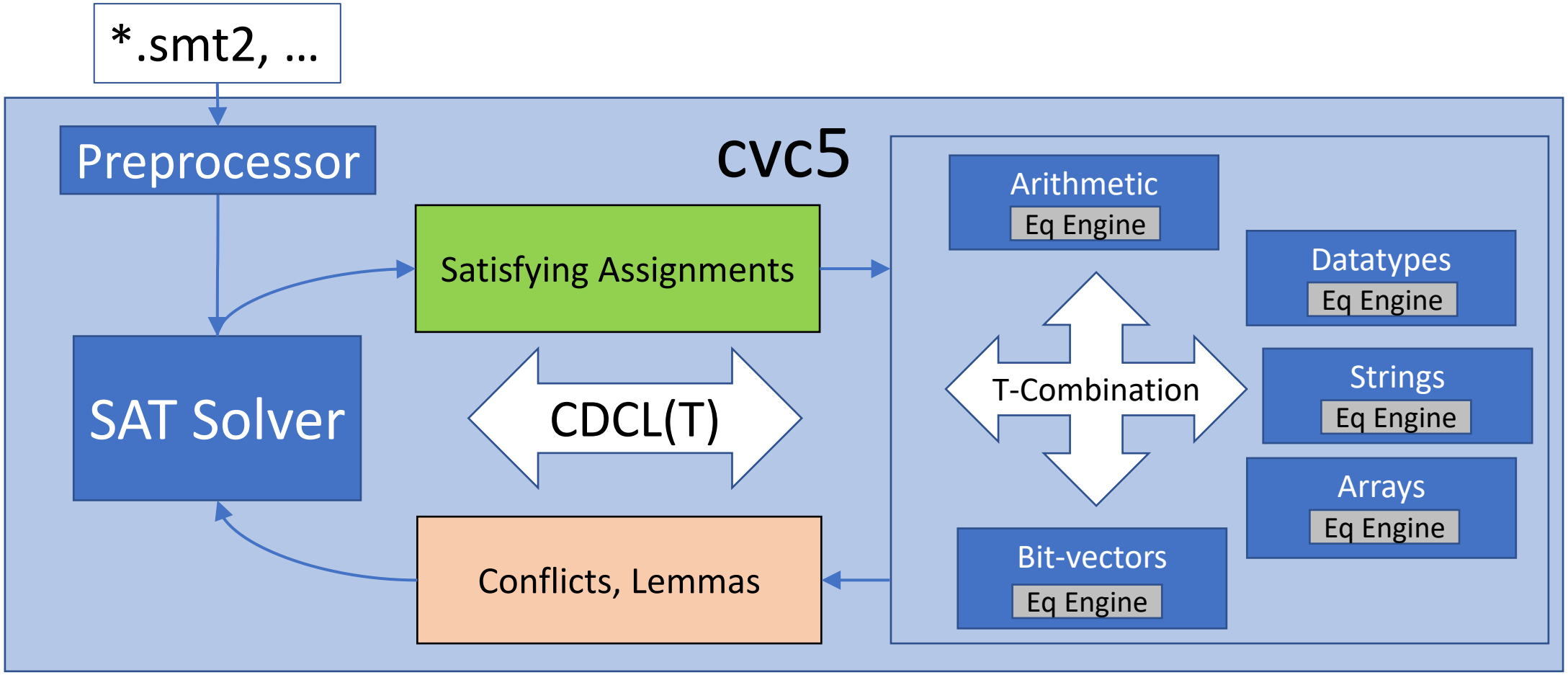


Architecture of cvc5



- Centralized methods for combining theories (Nelson-Oppen, polite)

Architecture of cvc5



- Eq reasoning, interface to T-combination managed by an *equality engine*
  - Standardized, mandatory for theories

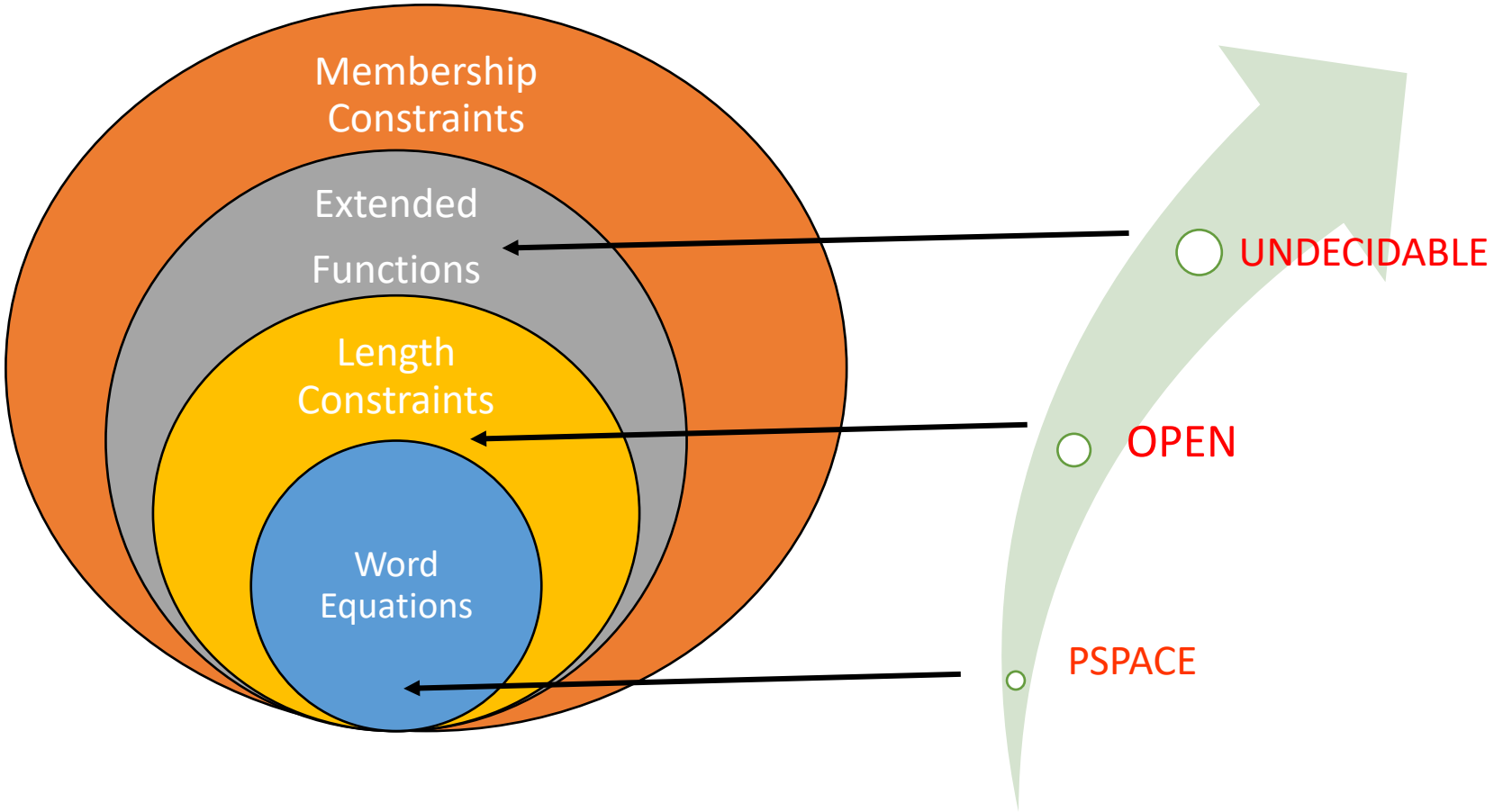
Design of Theory Solvers in cvc5

- Kinds file and type checker
  - Defines the signature of the theory T
- Rewriter
- Theory solver
  - Maintains an equality engine
    - Manages theory combination, T-propagation
  - Given calls to check a set of T-literals $\mathtt{M}$:
    - Return a subset of $\mathtt{M}$ that is T-unsat (**conflict clause**)
    - Return a T-valid **lemma**
    - Return a **model** for the variables of $\mathtt{M}$

$\Rightarrow$ In this talk: the theory solver for strings and regular expressions

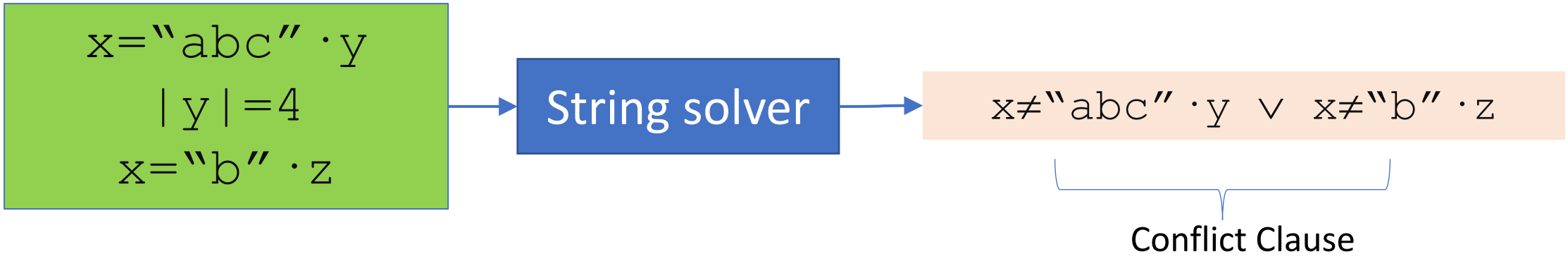
Designing the cvc5 Strings Solver

Strings and RegExp: Theoretical Challenges



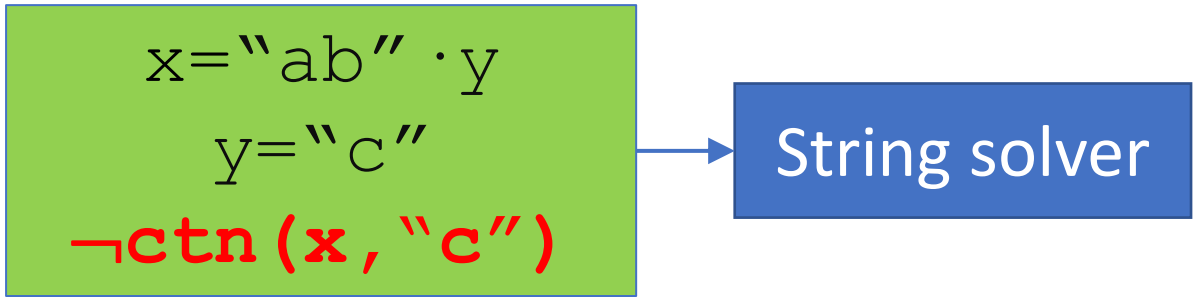
- Many applications require *extended string functions* and *RegEx memberships*
  - `ctn(x,"a"), to_lower(x)="abc", x∈range("A","Z")`

A DPLL(T) Theory solver for Strings [Liang et al CAV 2014]



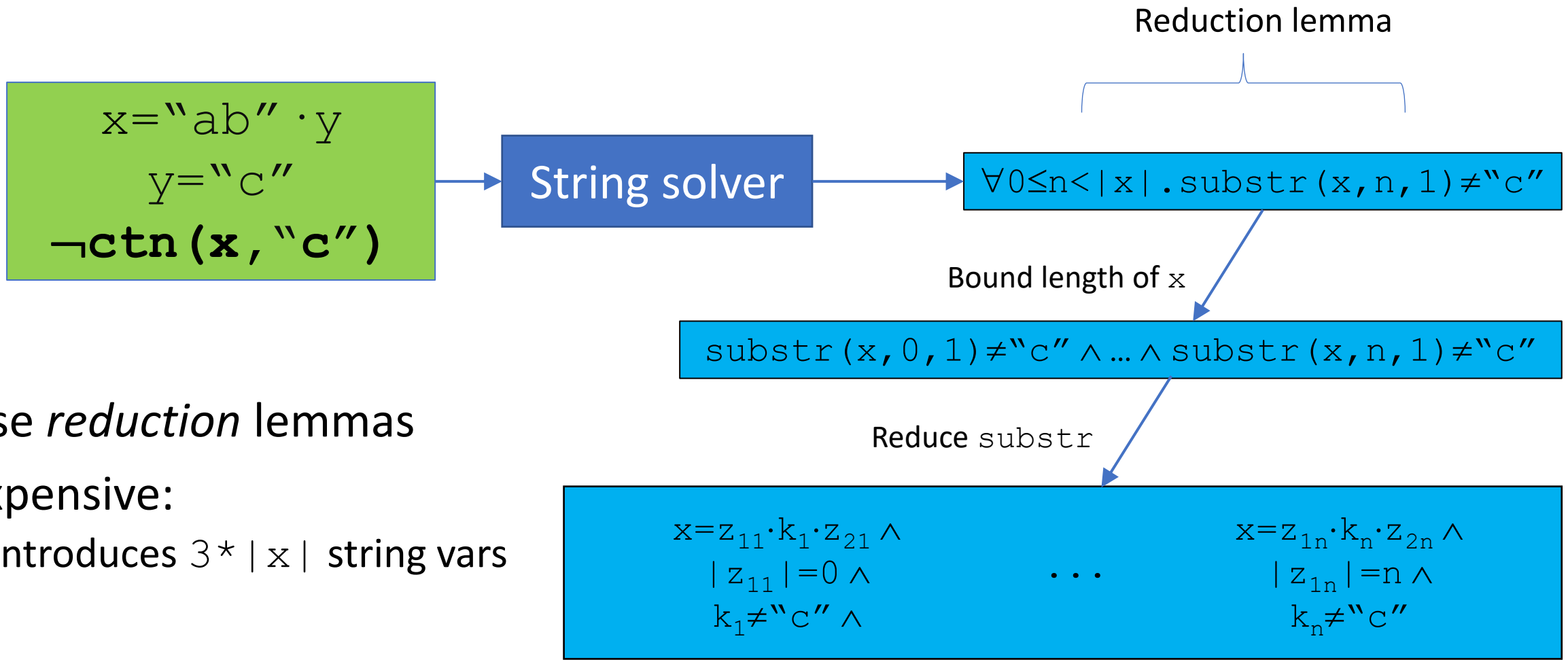
- A theory solver for a core theory of strings with concatenation and length
- Design a theory solver that is:
  - **Refutation and model sound** ("unsat" and "sat" can be trusted)
  - **Not terminating** in general
  - **Efficient** in practice

Extended Theory of Strings [Reynolds et al CAV 2017]



- Support *extended* string functions commonly used in applications
- For example: `ctn(x,"c")` denotes `x` contains the substring `"c"`

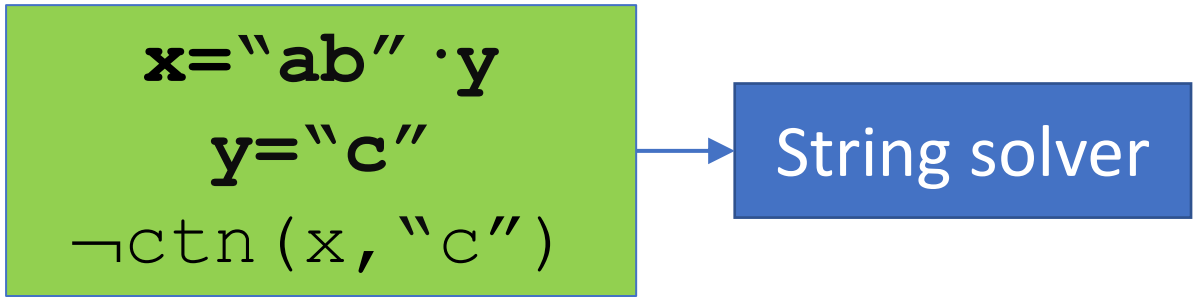
Extended Theory of Strings [Reynolds et al CAV 2017]



- Use *reduction* lemmas
- Expensive:

  Introduces 3*|x| string vars

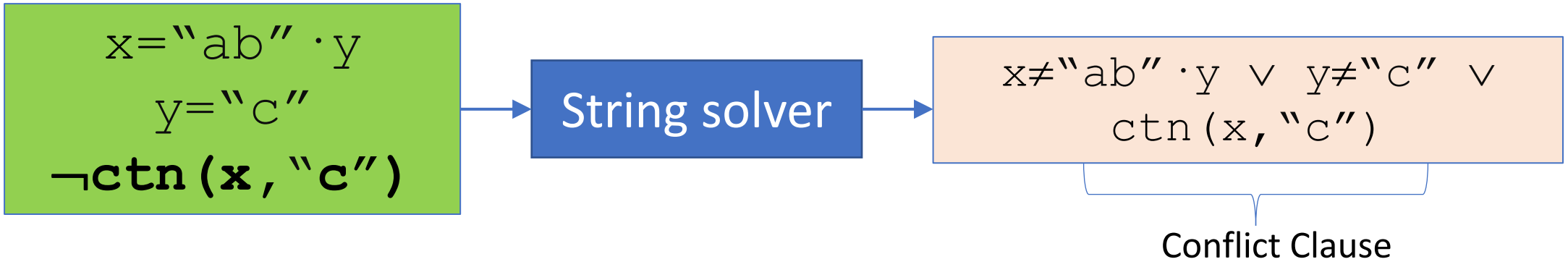
Extended Theory of Strings [Reynolds et al CAV 2017]



- Alternatively: use *context-dependent* simplification:

  $x="ab"\cdot y \wedge y="c" \models x="abc"$

Extended Theory of Strings [Reynolds et al CAV 2017]



- Alternatively: use *context-dependent* simplification:

  $x = \text{"ab"} \cdot y \wedge y = \text{"c"} \models x = \text{"abc"}$

  - Thus:

    $\neg ctn(x, \text{"c"})\{x \rightarrow \text{"abc"}\} \Leftrightarrow \neg ctn(\text{"abc"}, \text{"c"}) \Leftrightarrow \bot$

    By substitution — By rewriting

Recent Developments for Theory of Strings

- Context-dependent simplifications:
  - Highly aggressive rewrite techniques for strings **[Reynolds et al CAV 2019]**
  - Applied eagerly, integrated with equality engine **[Noetzli et al CAV 2022]**

- Reduction lemmas:
  - Improved encodings, witness sharing **[Reynolds et al FMCAD 2020]**
  - Model-based reductions **[Noetzli et al CAV 2022]**

- Broadening the core theory of strings:
  - String-to-code point (`code`) conversions **[Reynolds et al IJCAR 2020]**
  - Theory of sequences, support for `nth` and `update` **[Sheng et al IJCAR 2022]**

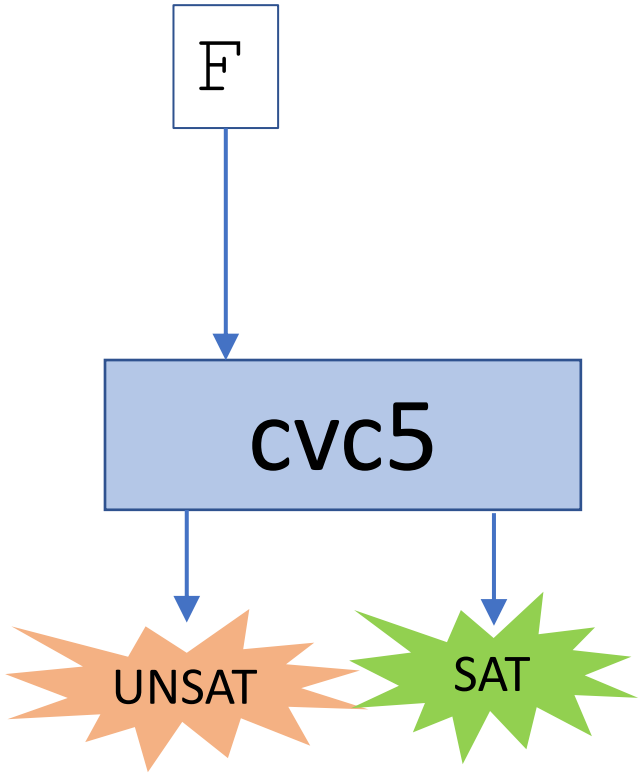
Designing Efficient Theory Solvers

- Use of standard engine for equality reasoning
- Cooperation with other theories
- Fast conflicts, context-dependent simplifications
- Lazy dependence upon expensive reasoning, e.g. reductions
- Other features not mentioned:
  - Proof support

Future Directions in cvc5

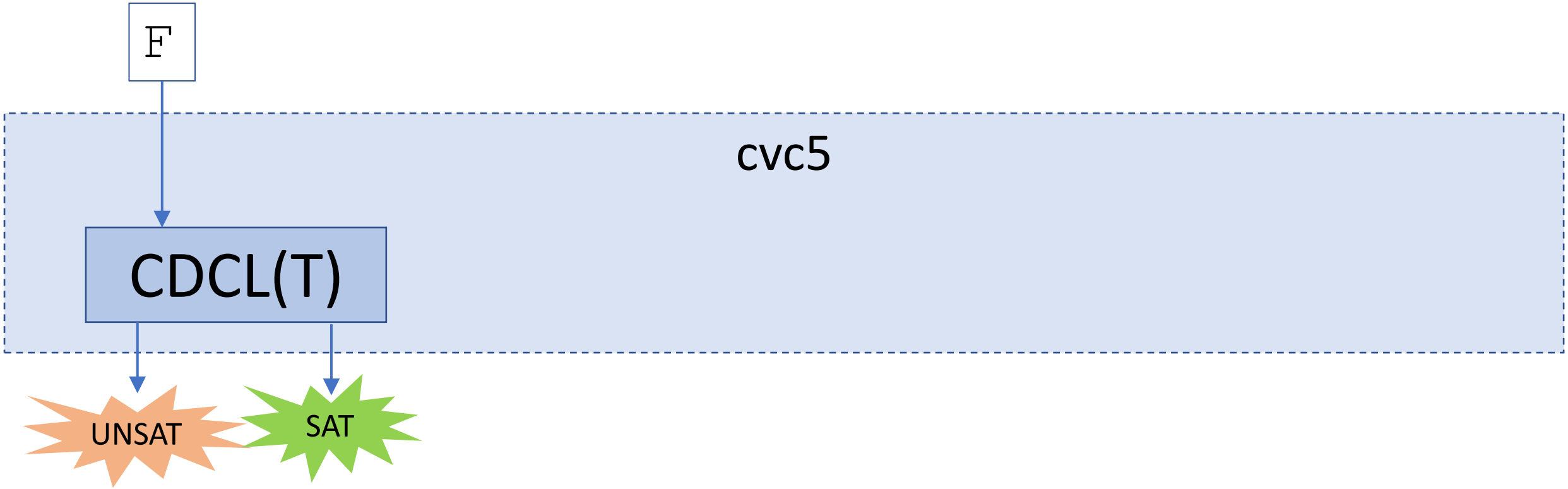
Advanced Architectures in cvc5

- What if we used the CDCL(T) engine as a black box?

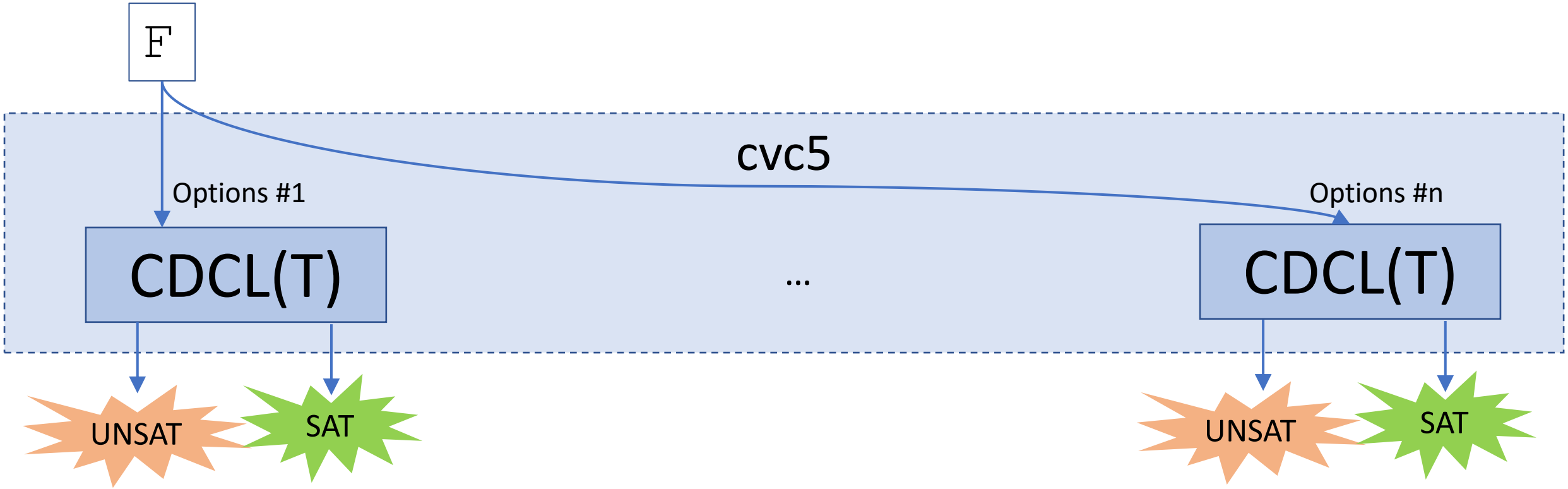


Advanced Architectures in cvc5

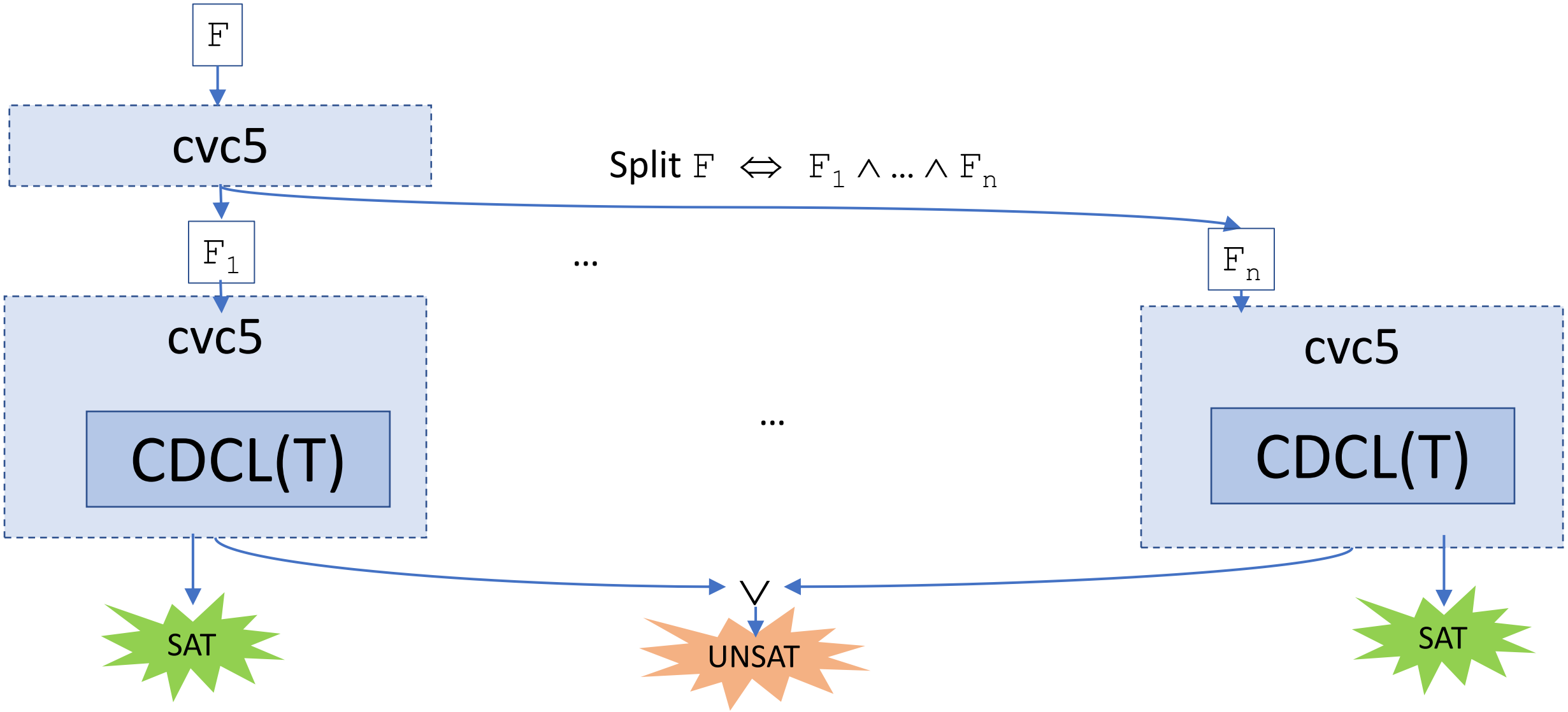
- What if we used the CDCL(T) engine as a black box?



Advanced Architecture: Portfolio

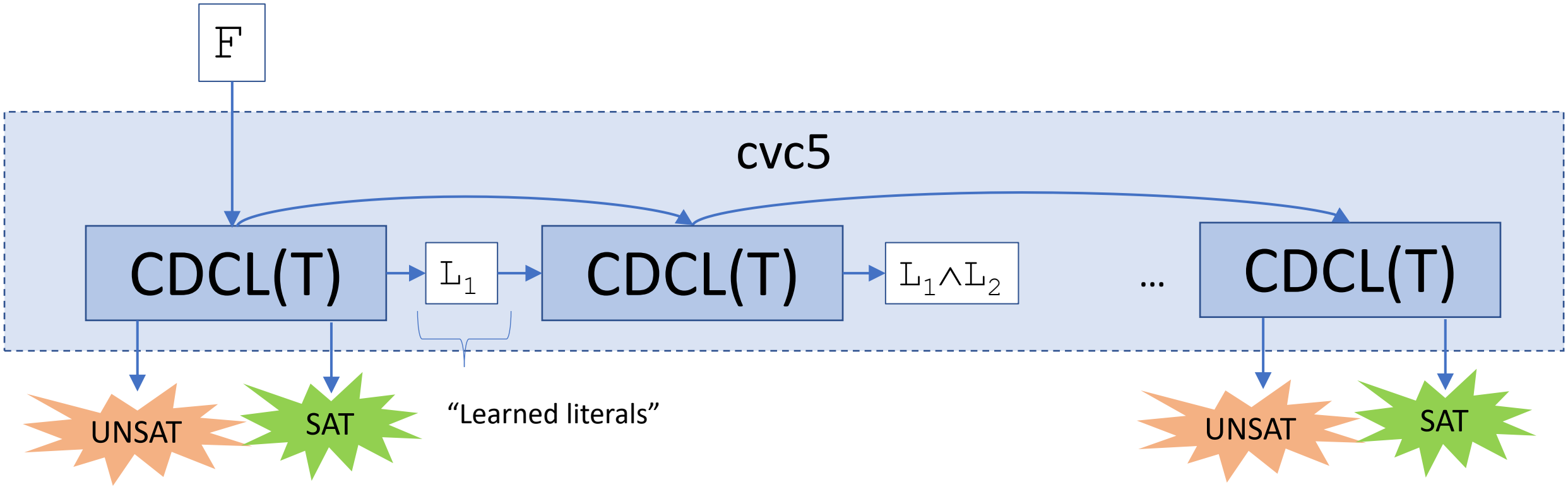


Advanced Architecture: Parallel



Advanced Architecture: Deep Restarts

- Idea: Restart after learning a set of literals that are implied by $\mathtt{F}$



Deep Restarts

- Given input formula F
  - A learnable literal l is:
    - Meets some syntactic criteria, e.g. l is a literal from F
    - $F \models_T l$
  - Can instrument the SAT solver to record learned literals
    - Literals l that are propagated at decision level zero

- A strategy for deep restarts:
  - The solver has learned at least one literal
  - No literal has been learned after some threshold (based on size of F)

- Learned literals may drastically impact preprocessing

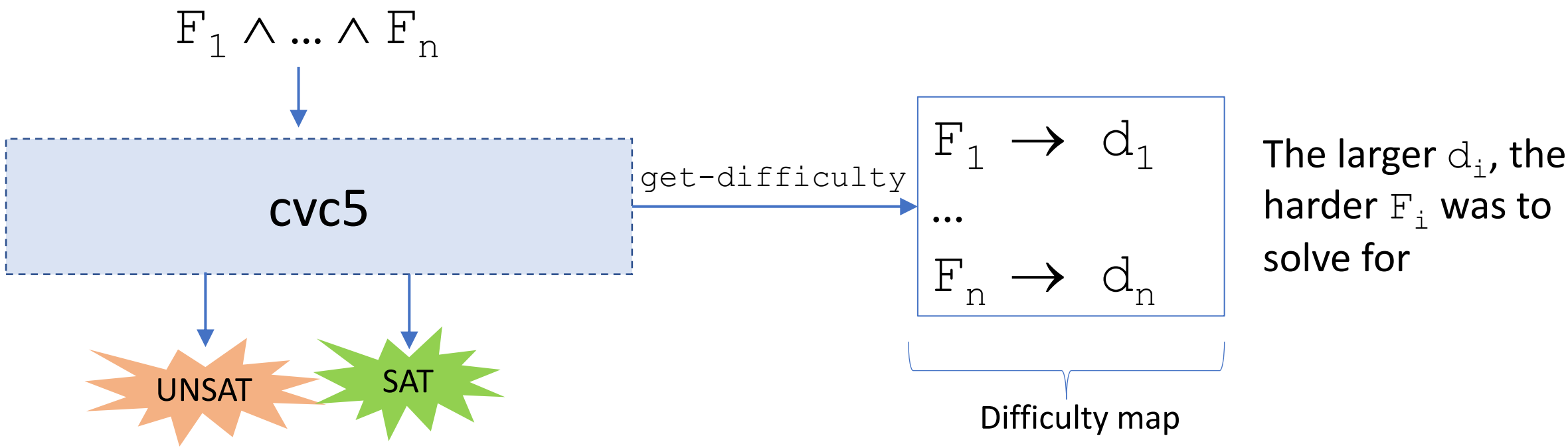
DEMO

Deep Restarts

- Possible variants:
  - In-processing: maintain SAT solver state?
    - Preprocessing changes mapping from SAT to theory literals
  - Restart while saving certain theory lemmas?
    - Based on usefulness criteria
  - Save to disk and restart later?

Difficulty Estimation

- When cvc5 can't solve an input, can we estimate *why* it was difficult?



Difficulty Estimation

- Given input $F_1 \wedge \ldots \wedge F_n$
  - Model-based:
    - When a candidate model M is constructed
      - Increment difficulty measure for each F_j that M does not satisfy
  - Conflict-based:
    - When a conflict clause ($l_1 \vee \ldots \vee l_n$) is raised
      - For each literal l_i, increment difficulty measure for the F_j s.t. $F_j \not\models \neg l_i$

DEMO

- Thanks for listening!