Symbolic Exploration

Azadeh Farzan

CS410 - Fall 2020

Reachability

- One of the simplest verification problems:
  - Given a set of bad states, can a program/system reach one of these states during its execution?
- It is a decision problem.

Propose a trivial algorithm for reachability...

Depth First Search!

State Space Exploration

- A program with 100 Boolean variables can have up to 2^{100} different reachable states.
- Representing these individually is not feasible.
- Symbolic representation accommodates representing sets of states more compactly.

Formally ...

A boolean formula F represents the set of all states s where

$$s \models F$$

$$states(F) = \{ s \mid s \models F \}$$

Logical Connectives as Set Operations

Let $S_1 = \text{states}(F_1)$ and $S_2 = \text{states}(F_2)$

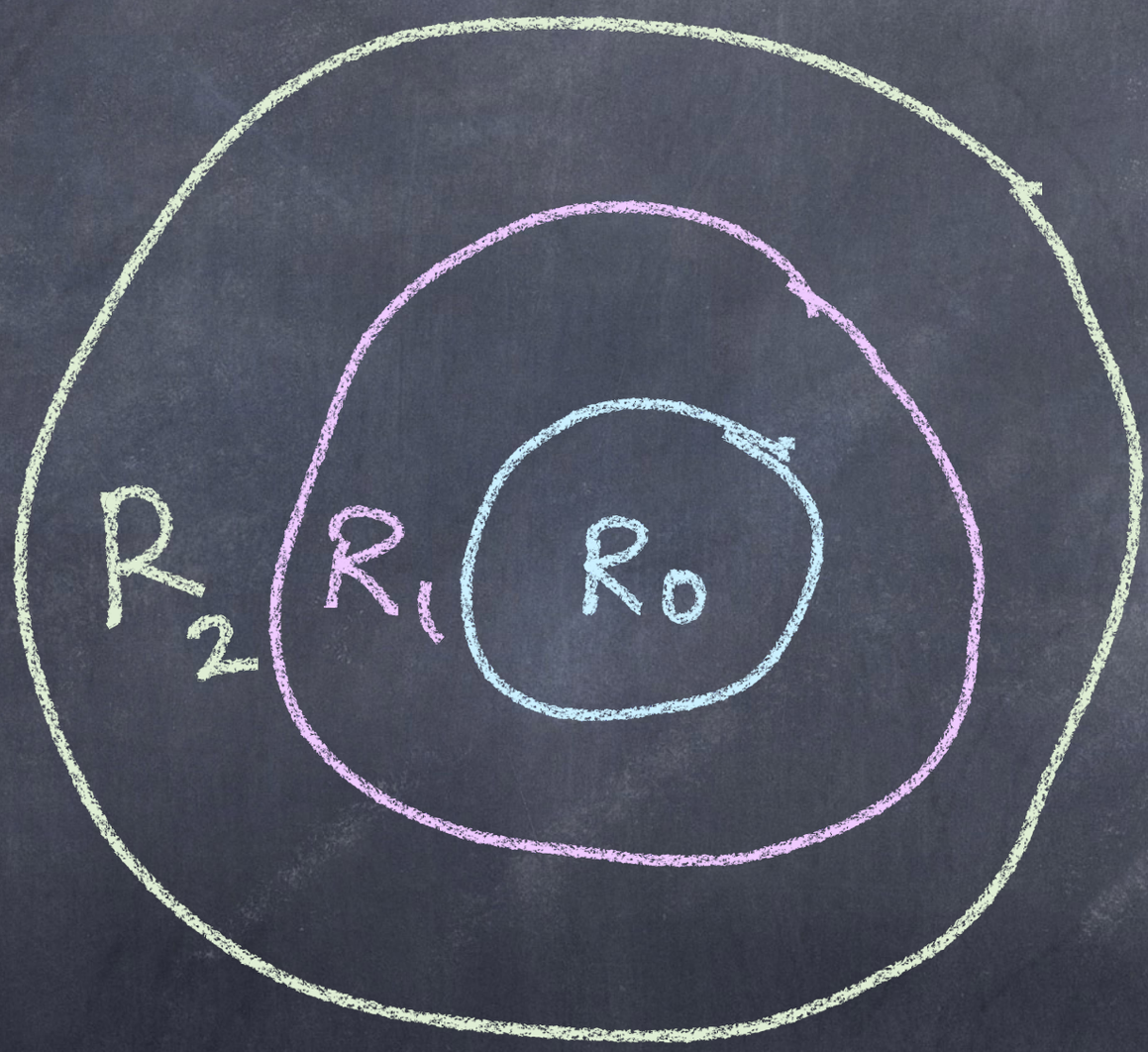
$S_1 \cup S_2 = \text{states}(F_1 \vee F_2)$

$S_1 \cap S_2 = \text{states}(F_1 \wedge F_2)$

$\overline{S_1} = \text{states}(\neg F_1)$

How do we solve reachability symbolically?

High Level Idea



$s_0 \in R_0$

↓ step

$s_1 \in R_1$

↓ step

$s_2 \in R_2$

step($\vec{v}$, $\vec{v}'$): a formula over two copies of state

First Attempt

$R_0 = I \longrightarrow$ set of initial states

$R_1(\vec{v}) = [R_0(\vec{v}') \wedge step(\vec{v}', \vec{v})] \vee R_0(\vec{v})$

reachable by one step from R_0

union R_0

$R_2(\vec{v}) = [R_1(\vec{v}') \wedge step(\vec{v}', v)] \vee R_1(\vec{v})$

$\vdots$

Let $E(\vec{v})$ represent all error states.

intersection of two sets

At each step j, if $R_j(\vec{v}) \wedge E(\vec{v})$ is satisfiable, then an error state is reachable.

$R_0 = L, R_1, R_2, R_3, \ldots$

If the system is <u>finite-state</u> then

$\exists j : R_j = R_{j+1}$.

We either find a j s.t $R_j \wedge E$ is <u>satisfiable</u> or a j s.t. $R_j = R_{j+1}$.

Let's build a better reachability algorithm!

Property-Directed Reachability

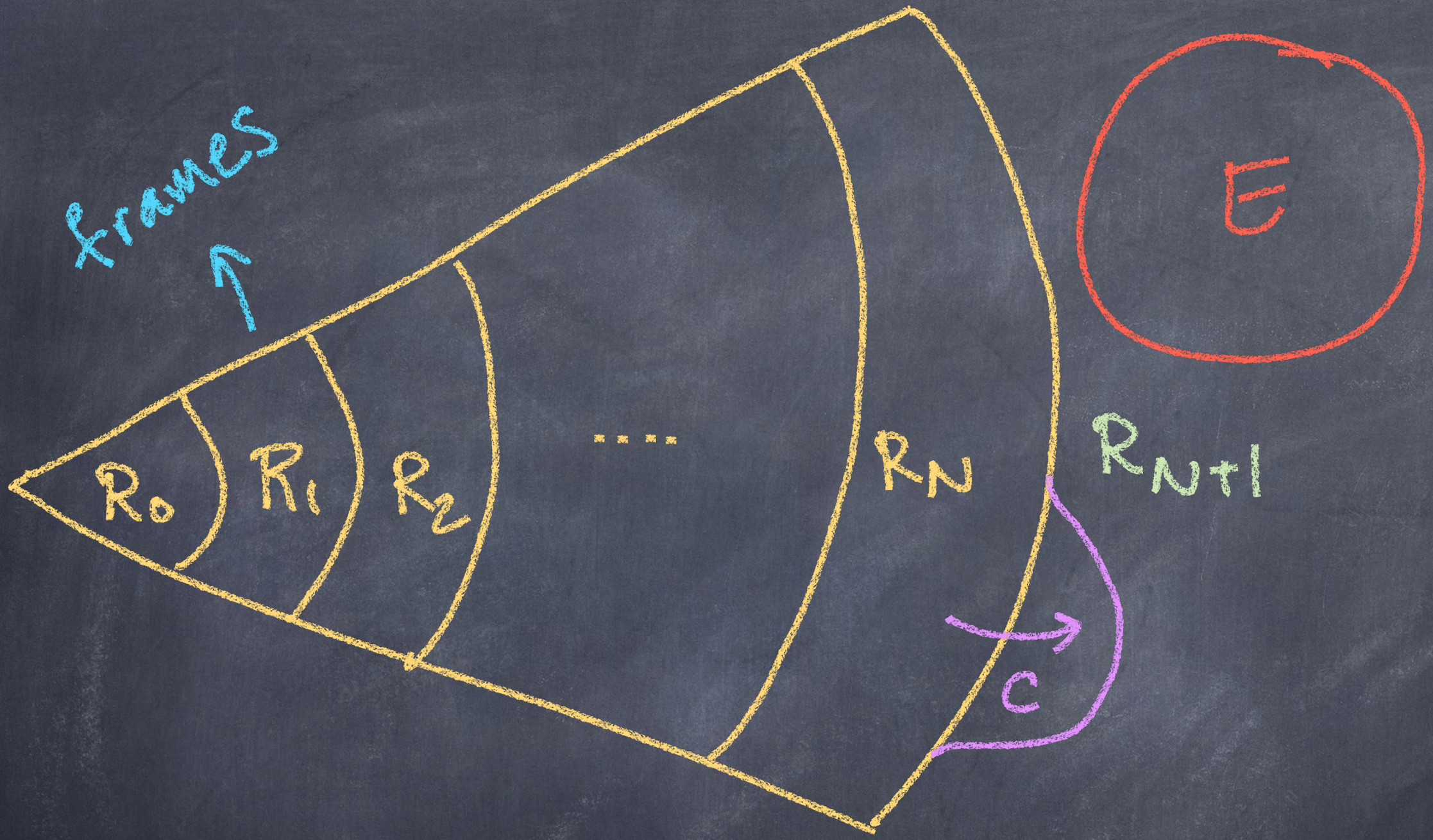
Setup

- Clause: disjunction of literals
- Cube: conjunction of literals

- Each frame R_j is a CNF formula.
  - But now, it is an over-approximation of the set of reachable states in j steps.
  - $CL(F)$: set of clauses in CNF formula F.

Invariants

- $R_0 = I$
- $\begin{cases} R_j \subseteq R_{j+1} \\ CL(R_{j+1}) \subseteq CL(R_j) \end{cases}$ $(j > 0)$
- $T(R_j) \subseteq R_{j+1}$ $\longrightarrow$ T: short for step
- $R_j \subseteq \neg E$ $\longrightarrow$ except the last frame N

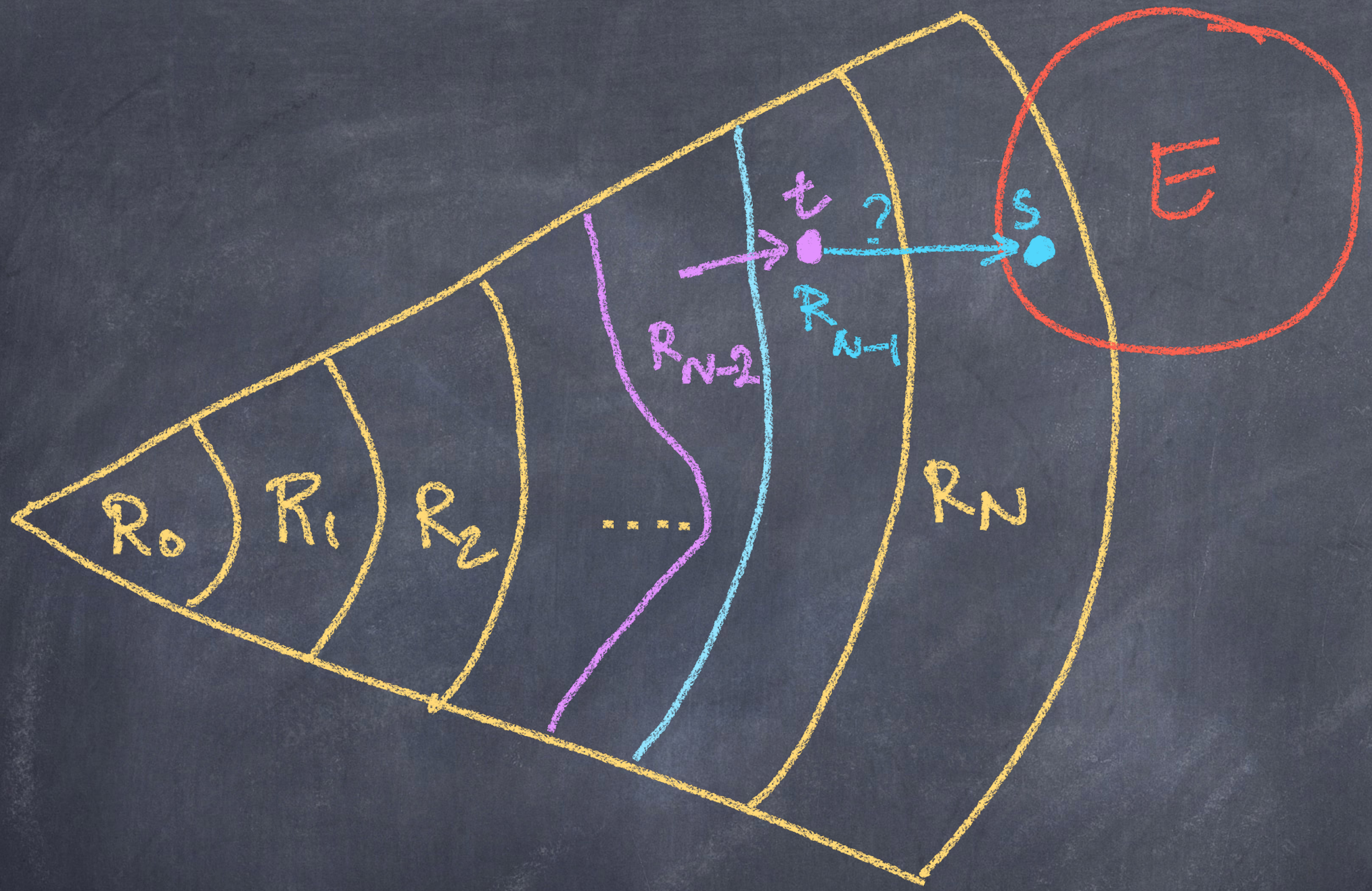


$R_N \wedge T \wedge \neg c'$ is not SAT

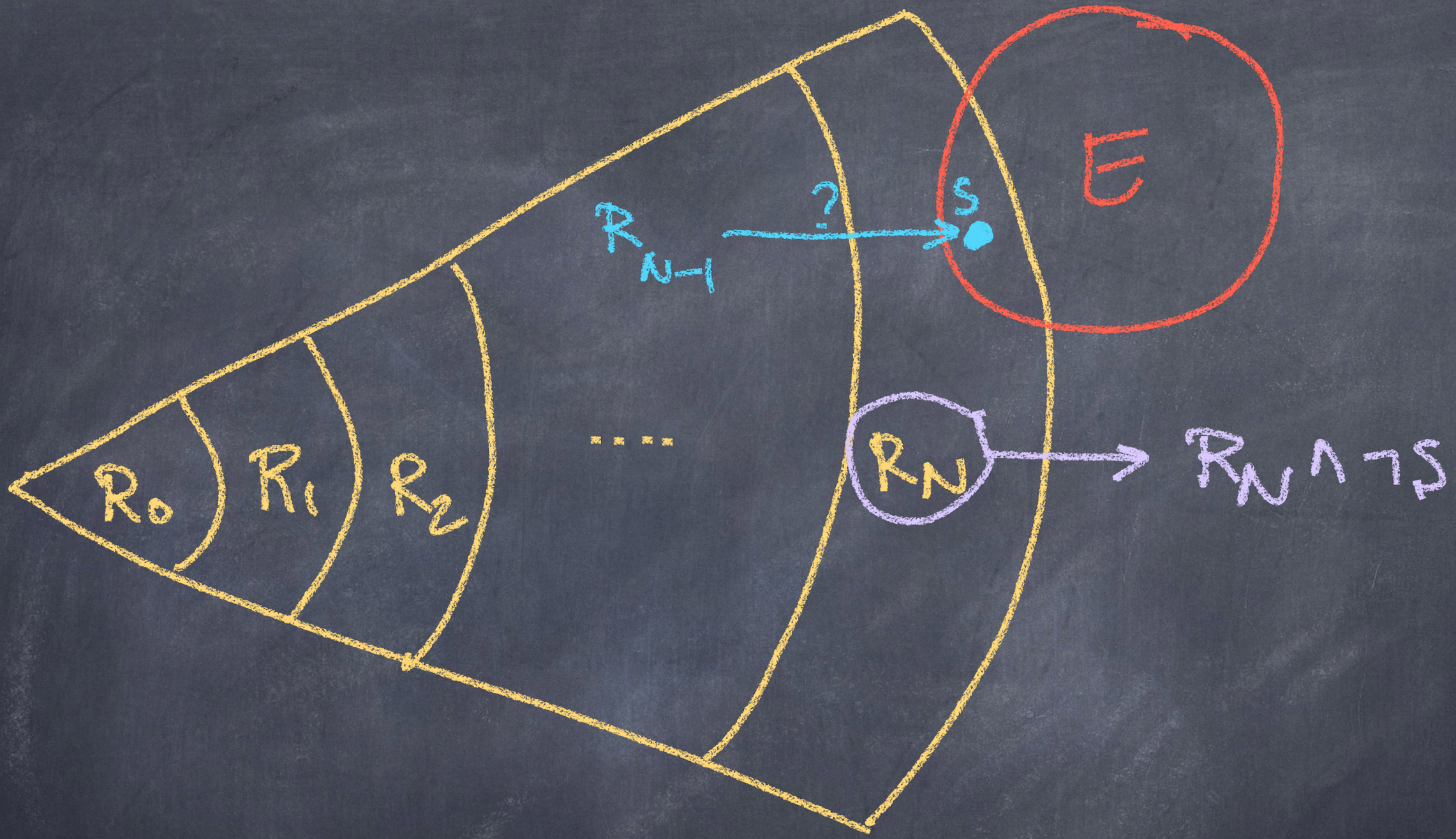
The Algorithm

Check if $R_N \wedge E$ is SAT.

- No? $R_N \subseteq \neg E$
  - new empty frame R_{N+1}
- $\forall j > 0$, Push clauses from R_j to R_{j+1}
  - clause $c \in CL(R_j)$ can be pushed if $R_j \wedge T \wedge \neg c'$ is not SAT.
- Terminate if two equal frames found



- is s truly reachable? Yes!
- is t reachable from the level before?



- is s truly reachable? NO!
- s is $\wedge$ cube $\Rightarrow \neg s$ is a clause

The Algorithm

Check if $R_N \wedge E$ is SAT.

- Yes? → careful: R_N was overshooting!
- There is a satisfying assignment $\underline{s}$.

(*) • Check if $R_{N-1} \wedge T \wedge s'$ is SAT

- No? Add $(\neg s)$ to R_N and start over
- Yes? get assignment t
  repeat step (*) with (R_{N-2}, t)

Wrap up