

Result-directed CHR Execution

Jon Sneyers
K.U.Leuven, Belgium

CHR 2010 workshop

Traditional CHR Execution

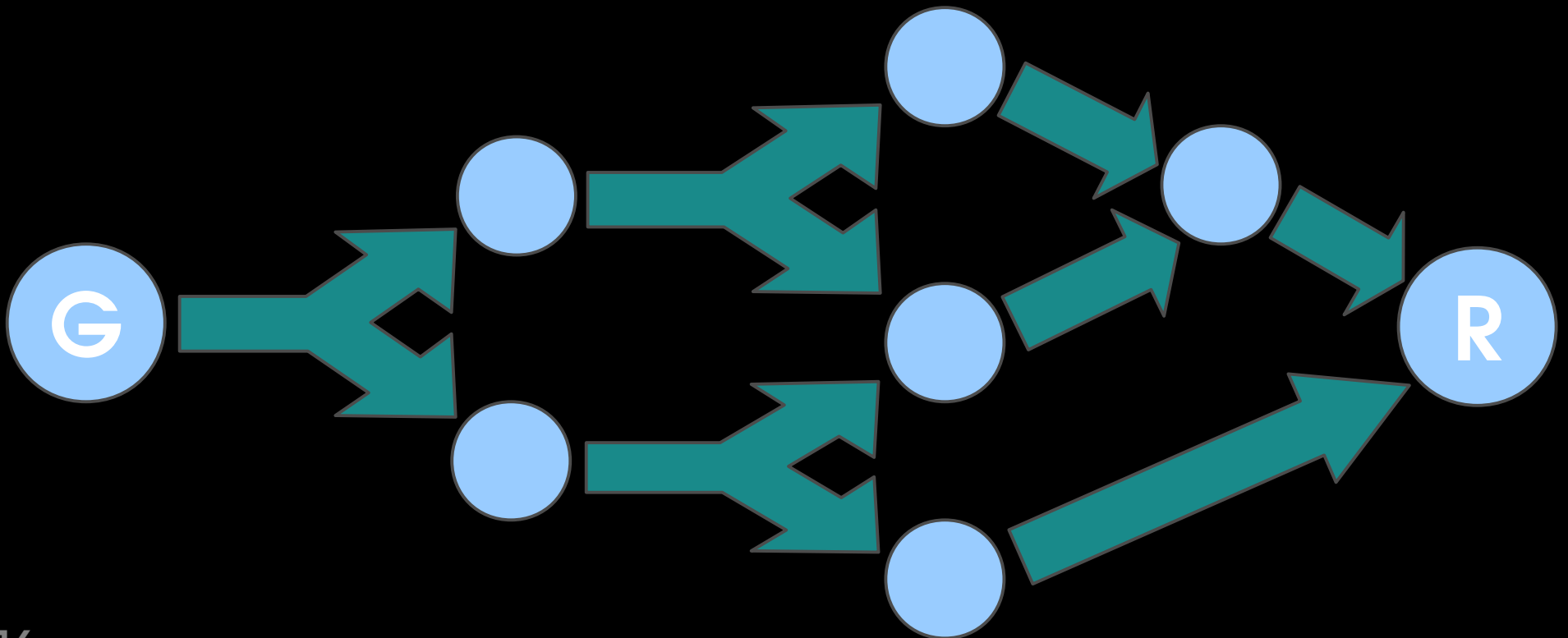
- Given a program **P** and a goal **G** (the query)
- Apply rules exhaustively to compute the result **R** (the answer)

Result-directed CHR Execution

- Given a program **P**, a goal **G**, and a result **R**
- Find a computation starting in **G** and ending in **R** (if it exists)

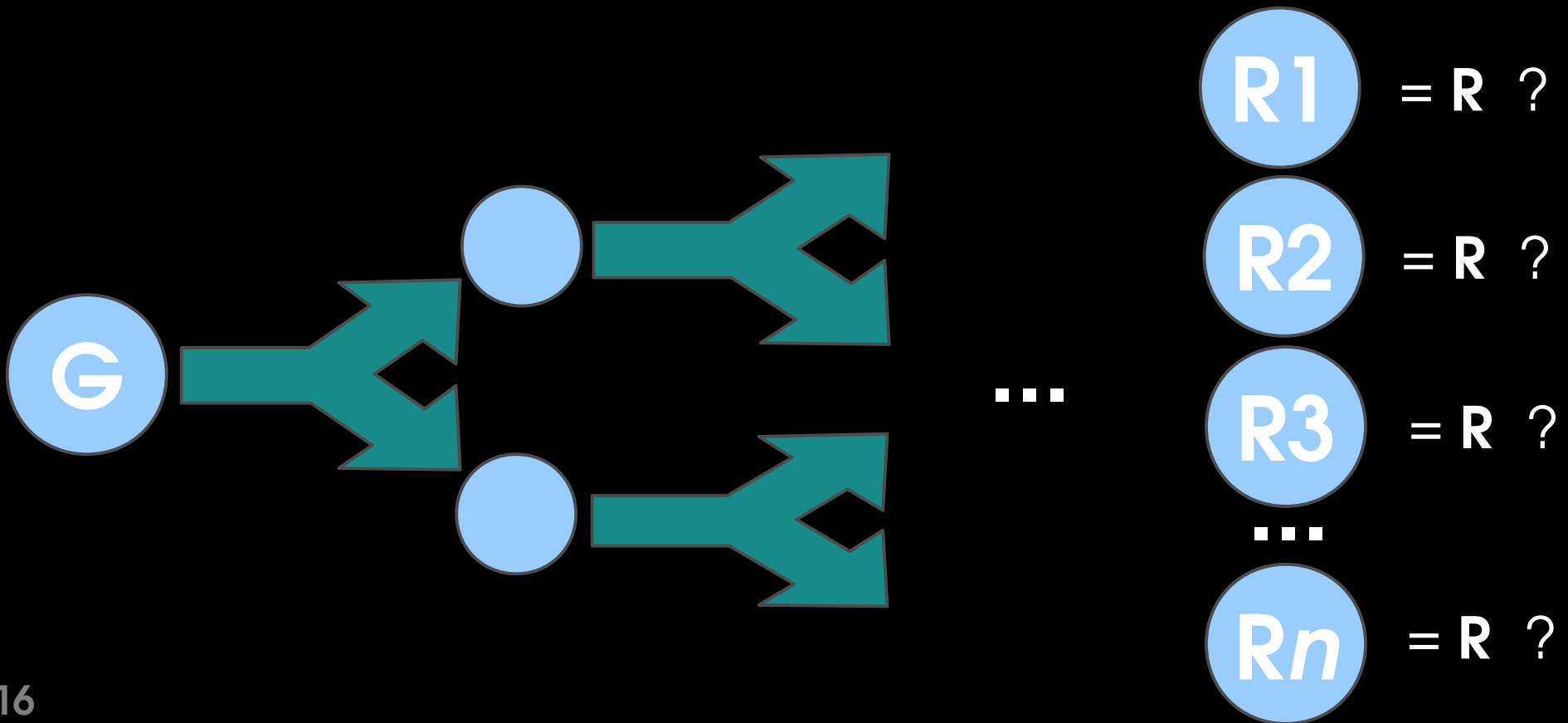
Confluent programs

- Easy to do for confluent programs
- If P is confluent, then just execute G and check whether the result is indeed R



Non-confluent programs

- Non-confluent programs may have a lot of results for one goal
- Computing **all** results could be too expensive

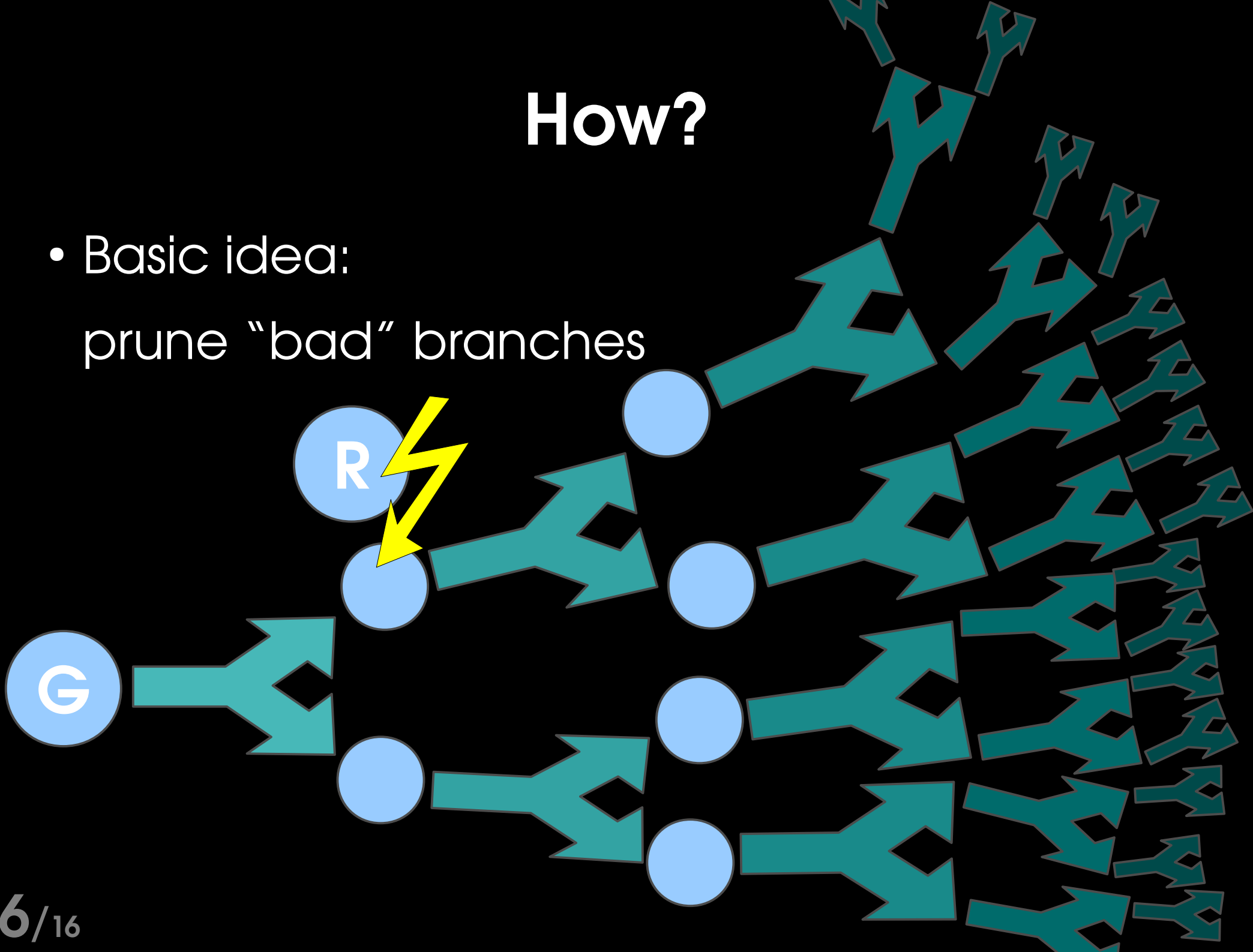


Why?

- Motivation: **CHRiSM** explanation search
- CHRiSM programs are non-confluent: the result depends on probabilistic choices
- Sampling is just traditional execution
 - Given a goal **G**, compute some result **R**
- Probability computation (and learning) involves result-directed execution
 - Given an observation **G** $\Leftarrow \Rightarrow$ **R**, what are the explanations? (sequences of probabilistic choices)

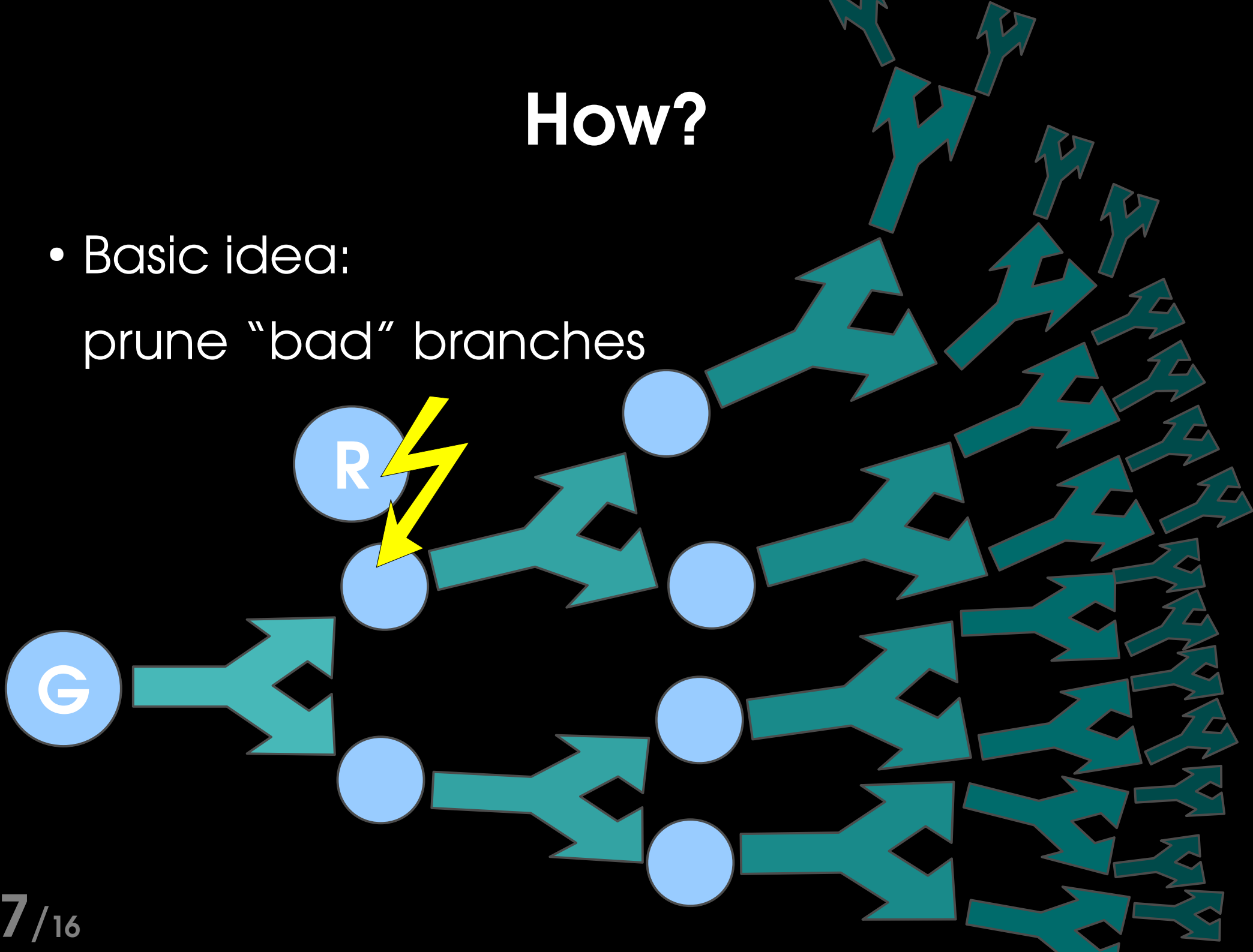
How?

- Basic idea:
prune “bad” branches



How?

- Basic idea:
prune “bad” branches



How to prune?

- Add “**early-fail**” rules to the program
- Such a rule detects that the intended result **R** can no longer be achieved, and fails
- This immediately prunes all further branches; host language starts backtracking
- Step 1: add result **R** to the goal as **result(R)**
 - $\text{result}((A,B)) \Leftrightarrow \text{result}(A), \text{result}(B)$
- Step 2: add early-fail rules

Early-fail rules (1)

- **Never-removed** constraint: only occurs in the *kept* head of rules

- For example: **c** is never-removed
 - Add these rules (in front of the program):
 - $c \implies \text{check}.$
 - $\text{result}(c) \setminus \text{check} \iff \text{true}.$
 - $\text{check} \iff \text{fail}.$
 - If the result does not contain **c**, then we immediately fail as soon as **c** is inserted.

- This is only sound for full observations (entire result **R** is given)

Early-fail rules (2)

- Take **functional dependencies** into account

- For example: **home(Person, City)** has a functional dependency **Person \rightarrow City** (and is never-removed)

→ Add the following rule:

$\text{home}(X,A), \text{result}(\text{home}(X,B)) \implies A \neq B \mid \text{fail.}$

- This is also sound for partial observations (only some subset of the result **R** is given)

Early-fail rules (3)

- **Surviving** constraint: may be removed, but it is always re-inserted or kept

- Examples of surviving constraints:

$\min(X) \setminus \min(Y) \Leftrightarrow X \leq Y \mid \text{true}.$

$\text{sum}(A), \text{sum}(B) \Leftrightarrow C \text{ is } A+B, \text{sum}(C).$

- Corresponding early-fail rules:

$\text{result}(\min(X)), \min(Y) \Rightarrow X \leq Y.$

→ If we know that sum/1 has positive arguments:

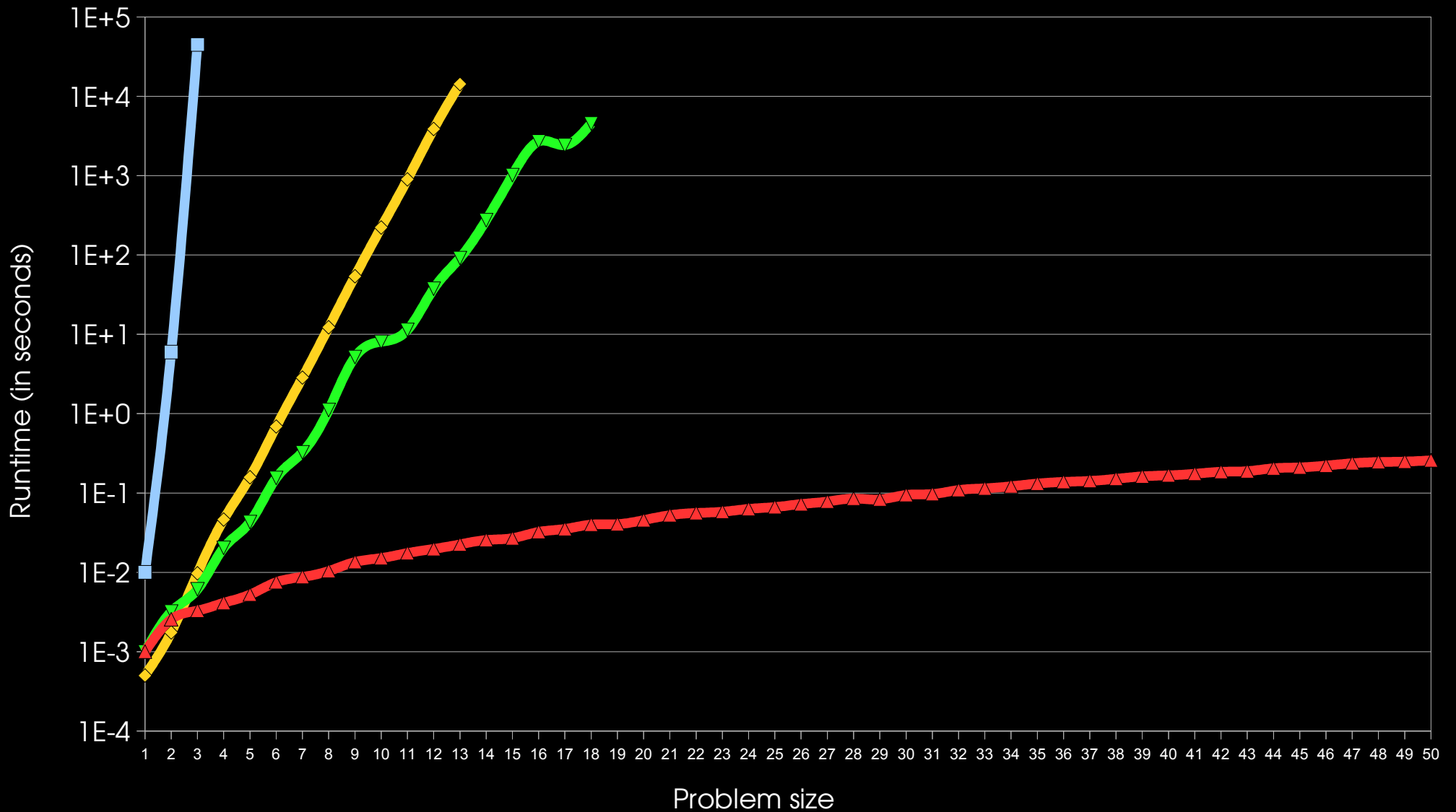
$\text{result}(\text{sum}(C)), \text{sum}(A) \Rightarrow C \geq A.$

Early-fail rules (4)

- User-defined early-fail rules
 - If you know specific properties of your program
 - e.g. something is always increasing/decreasing

Benchmark results

original Never-Removed NR+Surviving NR+S+User-defined



Practical results

The screenshot displays the APOPCALEAPS v0.3 software interface, titled "Automatic Pop Composer And Learner of Parameters". The interface is divided into several functional panels:

- Instrument settings:** Features "On/Off" buttons for Melody, Bass, Chords, and Drums. It includes "Resolution" dropdowns (16, 8, 8, 16) and "MIDI instrument" dropdowns (soprano sax, electric bass (pick), electric guitar (jazz)). It also has sliders for "Max jump (semitones)" (5, 17) and "Max repeat" (2, 5), along with an "Instrument range" section for c4 and c3.
- Global settings:** Includes "Key" selection (major/minor), "Time signature" (2/4), "Measures" (8), "Repeats" (4), and "Tempo (bpm)" (120).
- New song:** Contains a "Compose!" button, "Afterwards:" options (Play song, Show score), and "Play" and "Stop" buttons.
- Query selection:** Lists queries such as balad.q, default.q, hoempapa.q, ska-slow.q, ska-uptempo.q, and strings.q.
- Song selection:** A panel for selecting songs, currently empty.
- Style selection:** Shows "default.s" and includes an "Edit style..." button.
- Extra chaos:** A section with colored circles (yellow, blue, green, red) and a large grey circle.

A prominent yellow starburst graphic is overlaid on the "Compose!" button in the "New song" panel, indicating the primary action of the software.

Future work

- Implement automatic analysis to detect never-removed constraints, surviving constraints, etc.
- Invent new kinds of early-fail rules
- Early-succeed rules? (for partial observations)
- ... (see paper)

Conclusion

- Traditional CHR execution mode:
 - Given a goal, find a result
- New execution mode for CHR:
result-directed execution
 - Given a goal **and a result**, find a **derivation**
- Has applications in CHRiSM, CHR^v, ...
- Can be done efficiently by adding **early-fail rules** and then doing normal execution (with backtracking)