

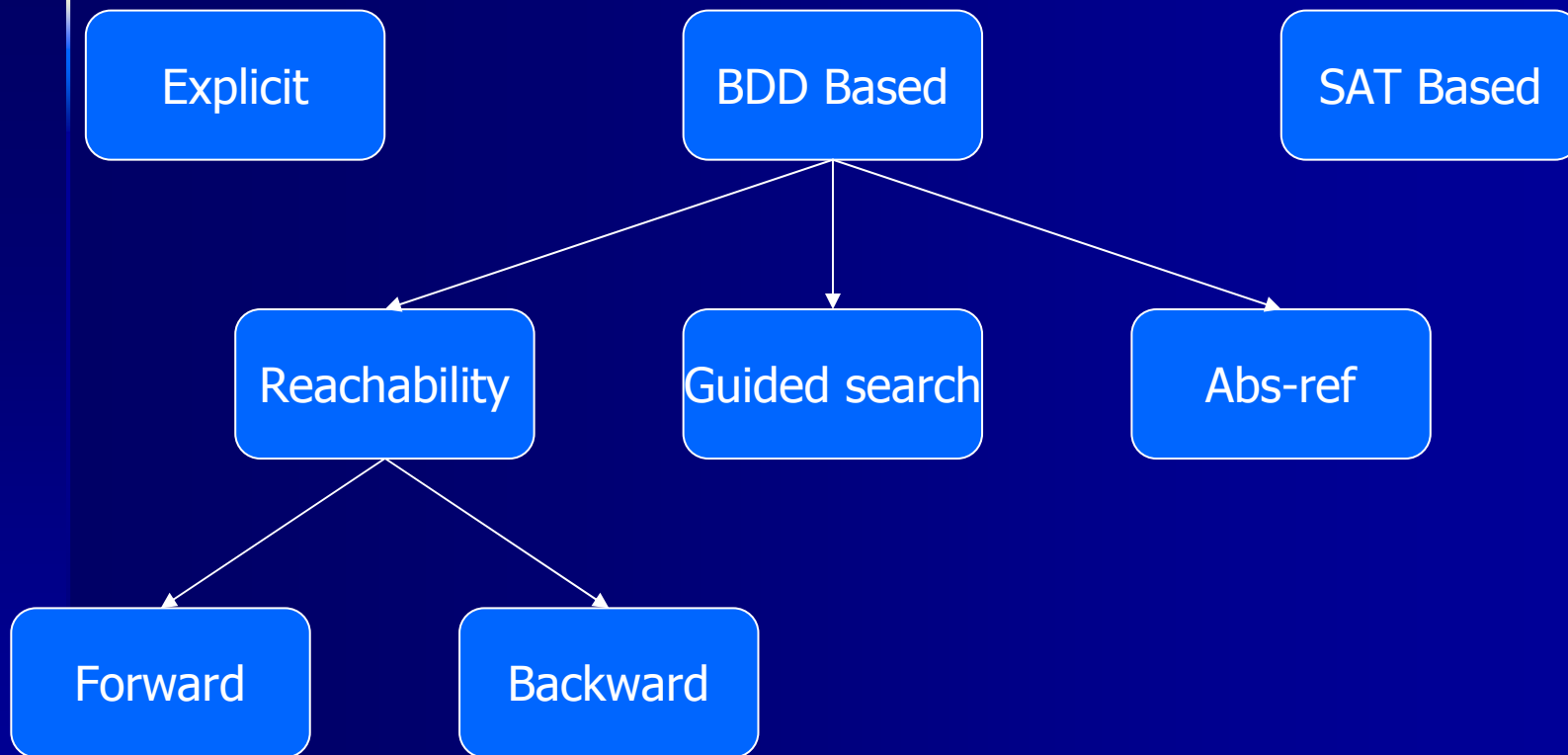
# **User-Friendly Model Checking: Automatically Configuring Algorithms with RuleBase/PE**

Ziv Nevo

IBM Haifa Research Lab

Haifa, Israel

# How many model checkers are there?



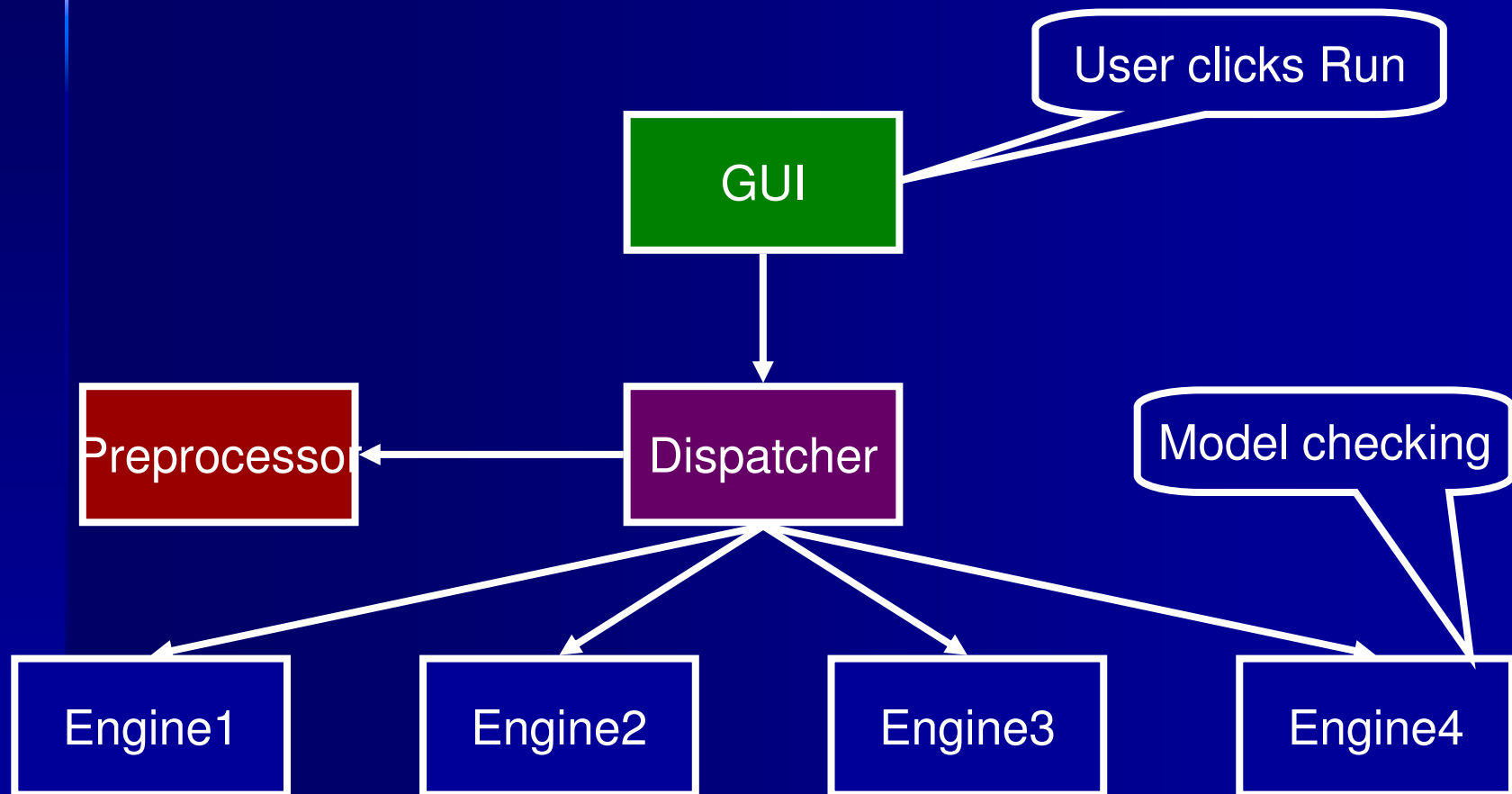
# Do we need all of them?

- Yes, or at least a decent collection
- There is never a clear winner for a PSPACE-complete problem
- The problem is even worse:
  - Each MC comes with a few settings, which may dramatically influence its efficiency

# So how do I choose?

- How about not choosing?
- Why not running them all in parallel, and let the best one win?
- This approach is known as **State Of The Art (SOTA)** model checker

# RuleBase Parallel Edition



# Do I have enough CPUs?

- Probably not
- You will have to limit your set of algorithms



# So how do I choose?

- No easy way to predict the fitness of a specific algorithm to a given problem
- Some rules of thumb though:
  - Problem size considerations
  - Verifiers vs. Bug hunters
  - Safety vs. liveness
  - Use what worked best yesterday

# Can you do it for me?

- Yes, we can



- Here comes Whisperer



# RuleBase/PE Whisperer

- An automatic decision system
- Generates a set of configured algorithms for a given MC problem
- Works under a concurrency limit
- Integrated into RuleBase/PE

# Cool, how does it work?

- Key idea is weighting expert advice
- Each engine is assigned an **advisor**



# Advisors

- C++ objects with a well defined interface
- Advisors generate **advice**:
  - Assigning values to engine settings +
  - An estimation of the engine's fitness for the current problem (from 0 to 5)
- Advisors should adjust fitness according to progress
- May access problem data (size, type, ...)
- May analyze past performance

# Whisperer main loop

```
while (problem is unsolved)
  newAdvice = get_new_advice()
  reevaluate_running_advice()
  terminate_poor_running_advice(newAdvice)
  run_promising_new_advice(newAdvice)
  sleep(zzz)
```

# Advice selection

- Whisperer iteratively picks an advice using a weighted random selection
- Weights are (normalized) advice fitness scores
- Until process limit is met

# Advice termination

- Whisperer may terminate a currently running advice if there exists a more attractive advice
- A biased random choice considering
  - Runtime
  - Score difference (running advice vs. new advice)

# Advisor reliability factor

- Advice fitness score is normalized using the advisor's **reliability factor**
- A reliability factor reflects an advisor's success in predicting advice fitness
  - Recorded over time
  - Increased when correctly predicts fitness
  - Decreased when prediction is poor

# Experimental Results

Example	Size (FFs)	SOTA (16 CPUs)	Whisperer (4 CPUs)	Trained Whisp. (4 CPUs)
A	755	157	173	168
B	4996	208	<b>333</b>	<b>192</b>
C	5158	383	462	<b>384</b>
D	386	70	158	203
E	550	314	337	344
F	2025	116	<b>184</b>	<b>142</b>
G	2391	74	1027	<b>84</b>
H	1418	190	<b>190</b>	<b>199</b>
I	15113	874	973	<b>882</b>
J	140	265	714	<b>265</b>



**Questions?**