HW Counter prediction using machine learning

Prediction on binary code

- Use machine learning to correlate static code features with a vector of HW counters for each **chunk** of code
- Identify the good/bad code parts
- Provide feedback to the code generator

Approach

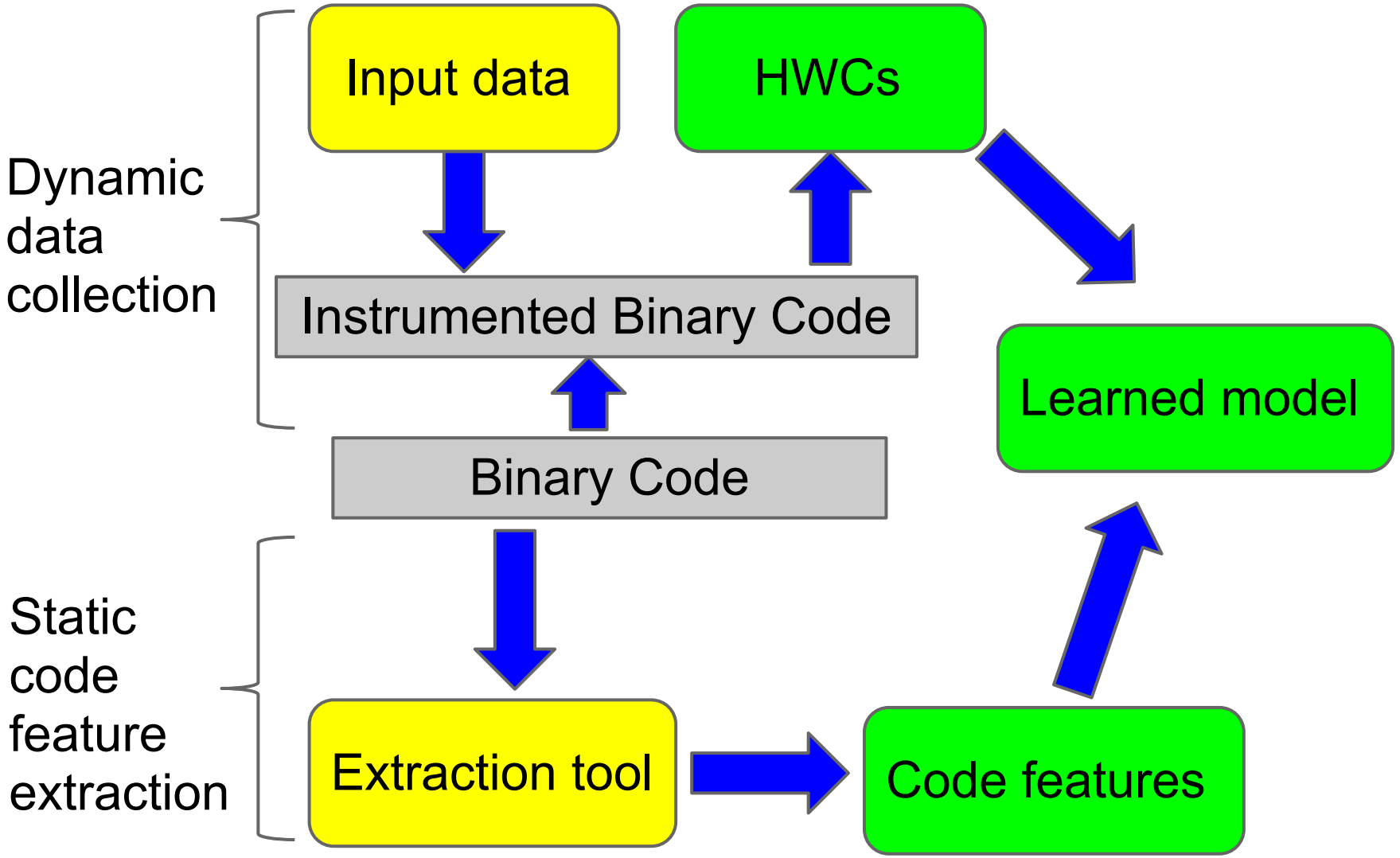
- Dynamic data collection
  - instruments each chunk of code to collect HWCs
  - serves as ground truth for training and validation

- Static code feature extraction
  - uses byte ngram, instruction ngram, graphlets
  - adds **addressing mode feature**

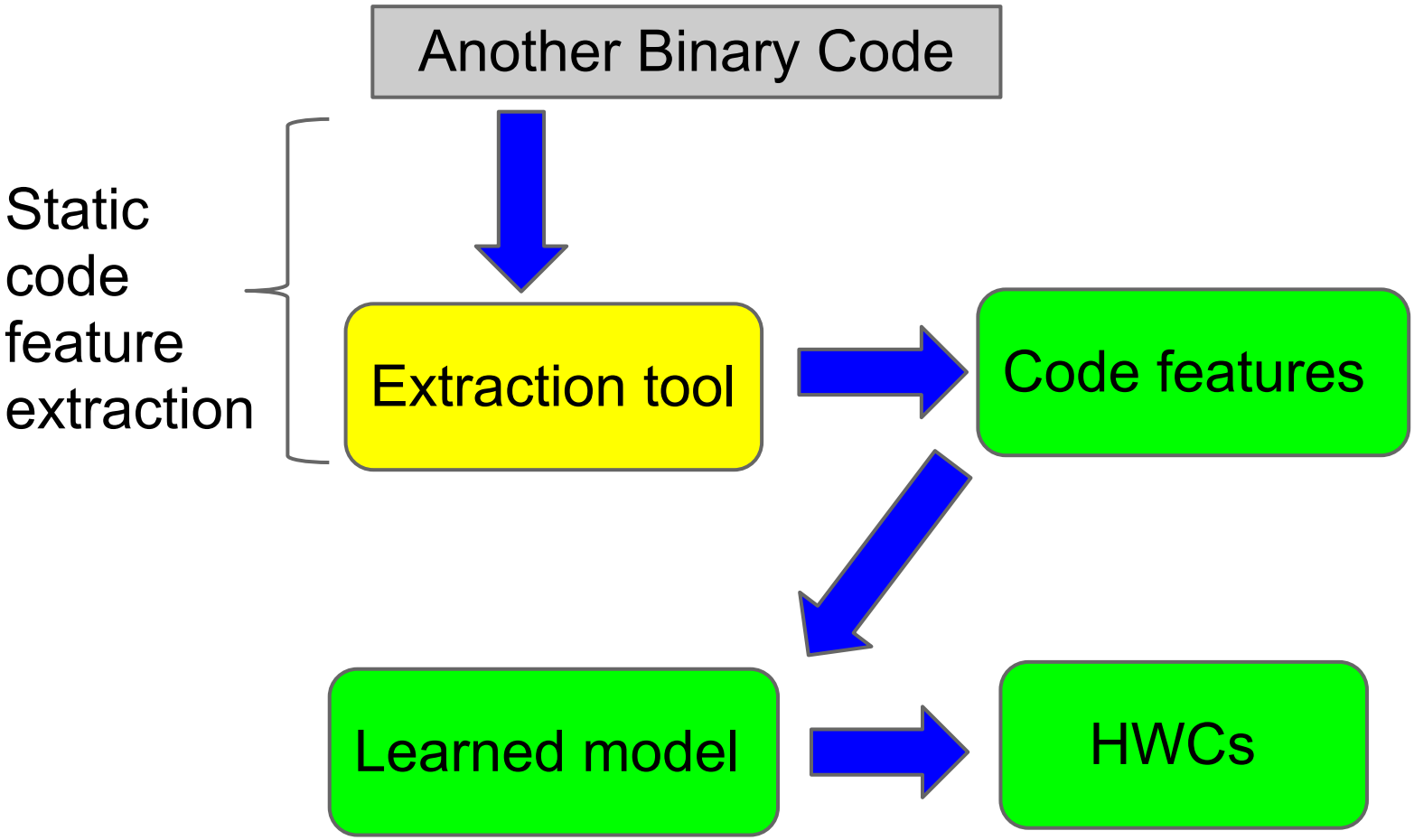
- Machine learning: use a learning toolkit

  scikit-learn: http://scikit-learn.org/stable/

Approach - Training



Approach - Prediction



How to define "chunk of code"?

- A good definition of "chunk of code"
  - makes it simple to collect HWCs
  - contains expressive static code features

- Basic block level
  - produces too huge data to store
  - disables graphlets (cannot capture control flow)

- Loop level
  - reduces to one piece of information per loop
  - enables graphlets to capture repeated code structure

Addressing mode feature

- It should capture the impact of data access on performance for different instructions

- Give scores to data access types

  Imm:1 point < Rax:2 point < [Imm]:3 point < [Rax]:4 point

- The score of an instruction is the sum of the scores of all operands

- Start with summarized score of a chunk

Status

- Loop dynamic instrumentation is done in Extrae
- Static code feature extraction works at function level
- To implement addressing mode features
- Select stream, matrix multiplication, and NPB3.2-SER as the first benchmarks

Conclusion

- The big picture is complete
- We do not know it works or not until we do experiment
- Prototype it