Synthesizing Self-Contained Scientific Software
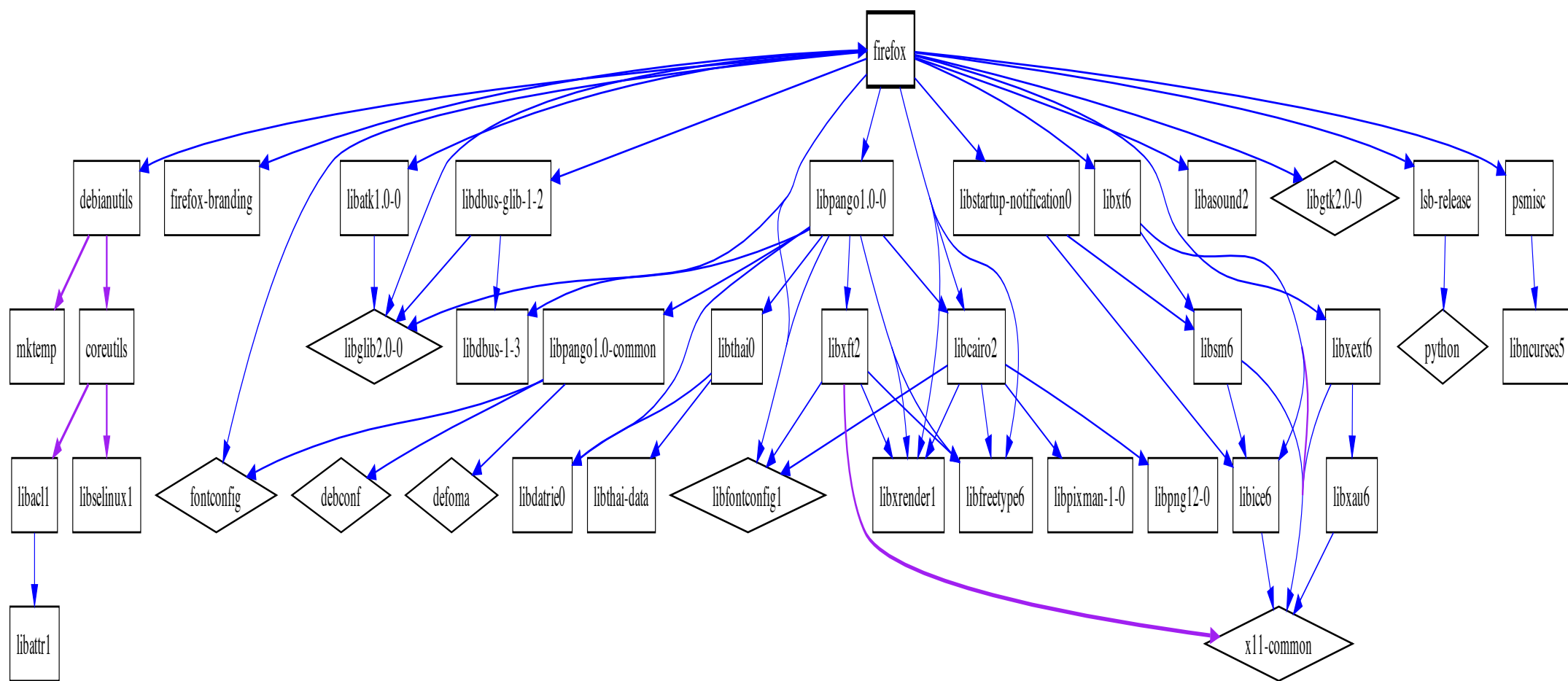
Ashish Gehani, SRI

Problem: Scientists use software to analyze their data. Result reproducibility depends on the ability to run the software. Changing environments, extraneous dependencies, and large data sets makes software distribution / reuse a challenge.

Preliminary Effort: Use package management to prune extraneous dependencies. Further refine using language-specific partial evaluation.

Use package metadata to compute set that can be removed.

Insert specialization directives for C-Mix/II, Tempo.

Current Approach: Tool to build self-contained software packages. Specify environment to minimize external dependencies. Use partial evaluation to winnow the codebase. Specialize with respect to target data to reduce package size.

```
> occam ./configure
> occam make
> occam make install
```
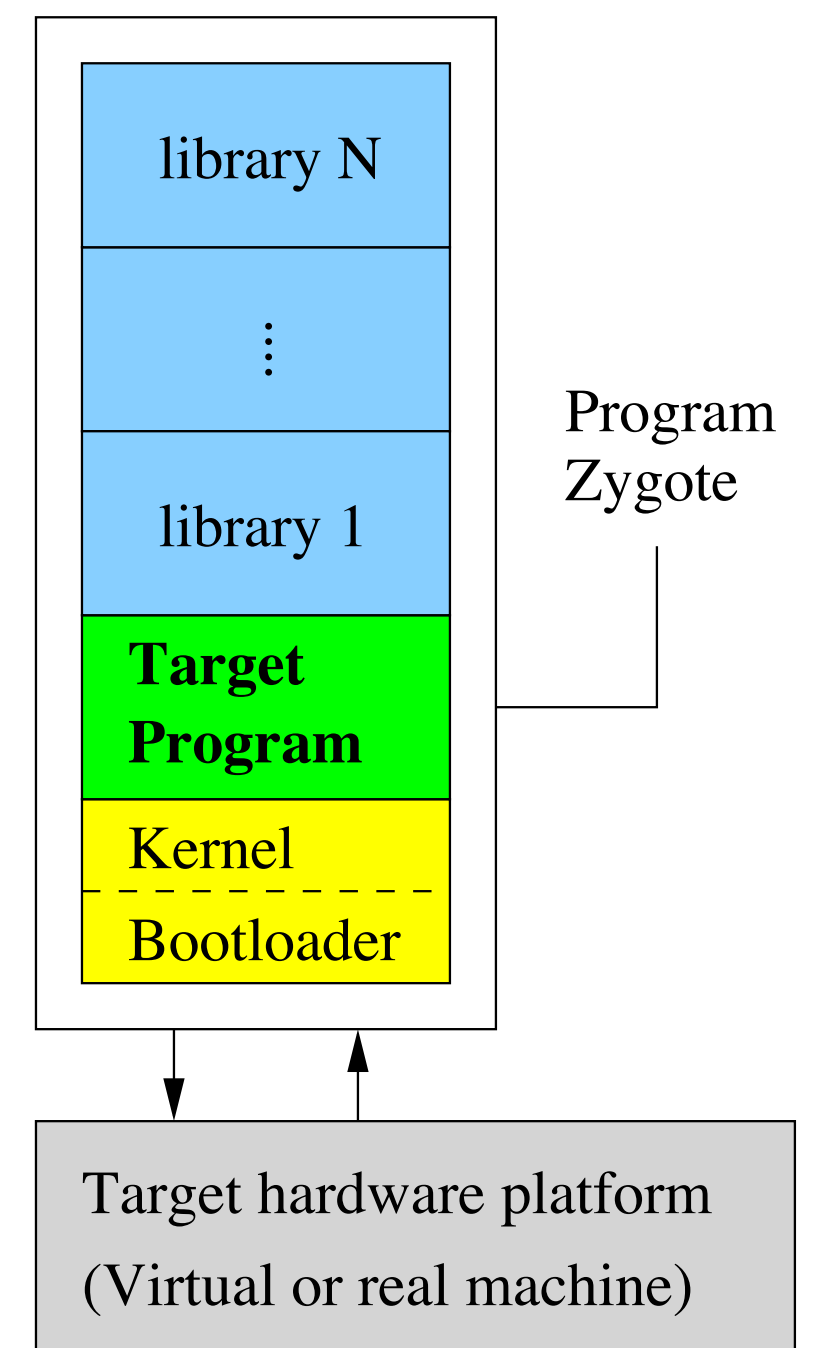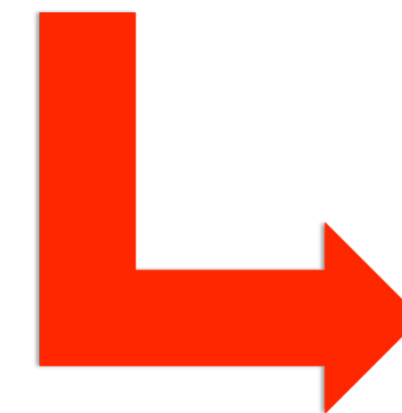

Build program with all code dependencies.

```
#include <math.h>
float compute_building_height(float building_distance){
        float viewing_angle = pi/4;
        float building_height =
                compute_opposite(building_distance, viewing_angle);
        return building_height;
}
float compute_opposite(float adjacent, float angle){
        float opposite = adjacent * tan(angle);
        return opposite;
}
```

Use partial evaluation with respect to known inputs to prune program, dependencies.

```
float compute_building_height(float building_distance){
        float building_height =
                compute_opposite_specialized(building_distance);
        return building_height;
}
float compute_opposite_specialized(float adjacent){
        float opposite = adjacent * 1;
        return opposite;
}
```

Use LLVM compiler optimization for intra-procedural pruning and iteratively compute inter-procedural interfaces (specializing when possible) till fixed point reached.