Finding Suitable Programs:

Semantic Search with Incomplete and Lightweight Specifications

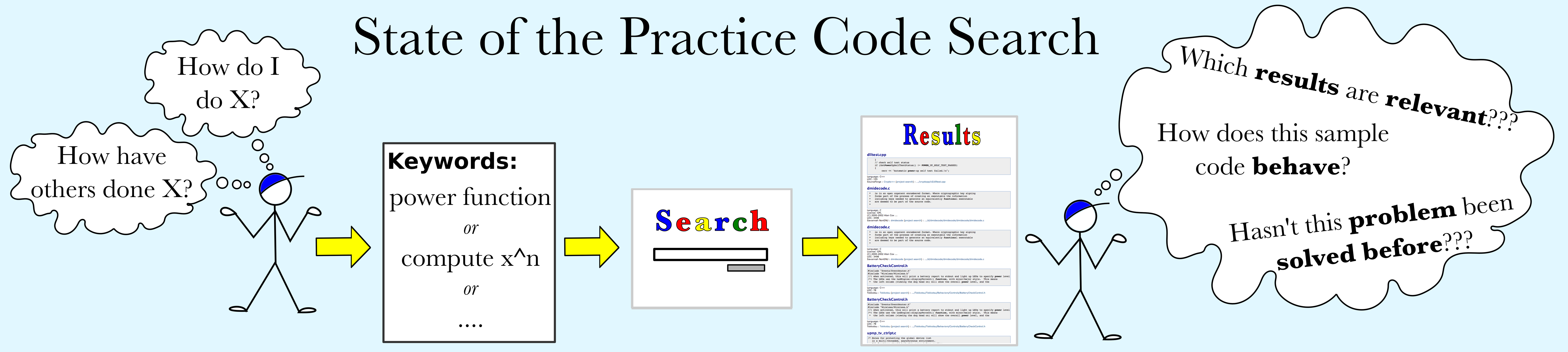
Kathryn T. Stolee

Advisor: Sebastian Elbaum

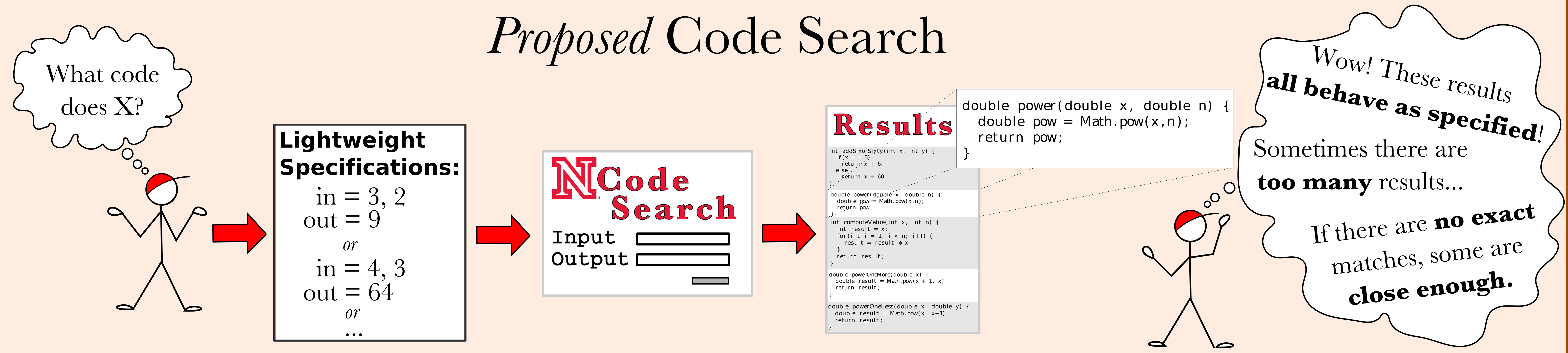
University of Nebraska-Lincoln

kstolee@cse.unl.edu - http://cse.unl.edu/~kstolee

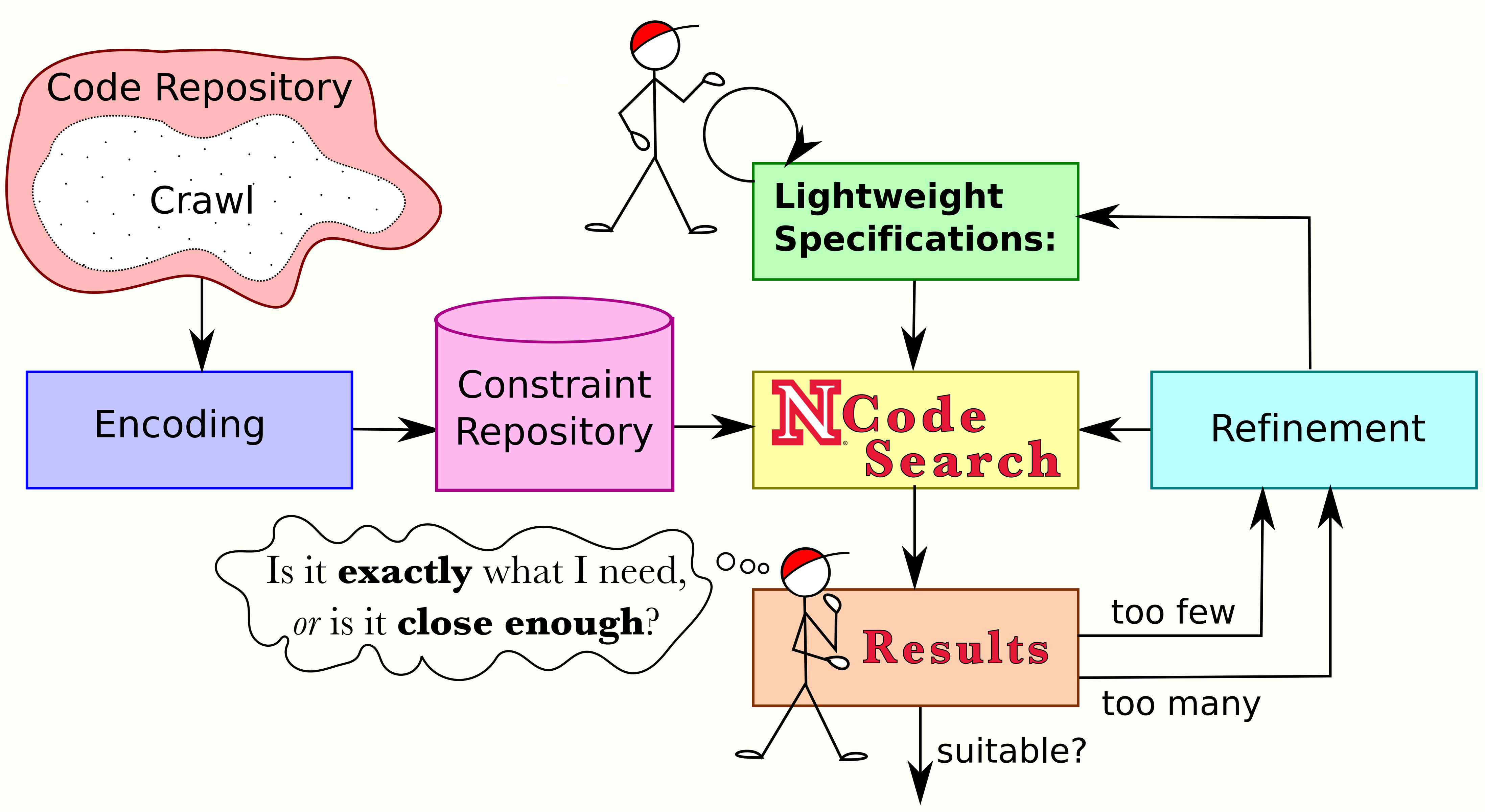
State of the Practice Code Search



Proposed Code Search



How does it work?



Code Repository. We leverage publicly-acccessible, large code repositories to helps programmers find relevant code, promote reuse, and increase productivity.

Targeted Domains: Yahoo! Pipes (next: SQL and Java)

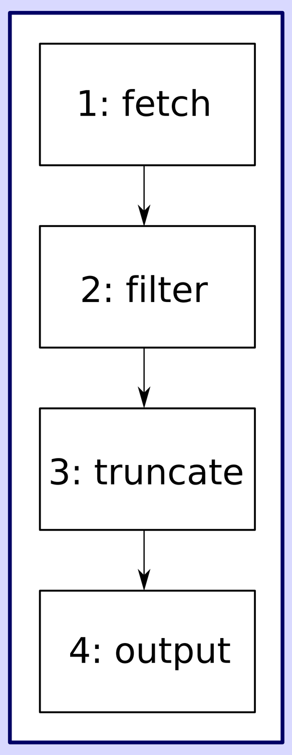
Lightweight Specifications. Instead of using textual queries, this approach uses lightweight and incomplete specifications to characterize the desired behavior of the code. The specifications are in the form of input/output pairs and/or partial program fragments.

For Yahoo! Pipes → **Input:** http://some.url/rss → **Output:** selected items

Constraint Repository. The programs in the code repositories are encoded as constraints, forming the constraint repository. An SMT solver searches this repository for matching programs.

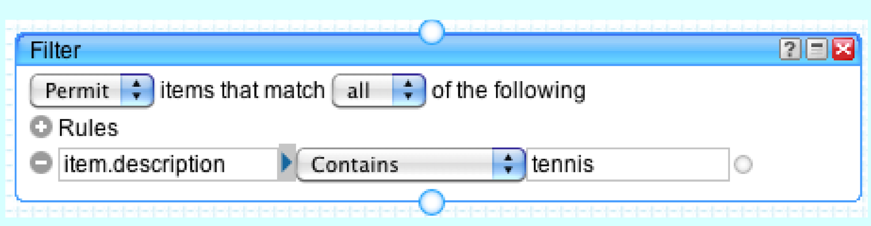
Code Search. To determine which of the programs in the constraint repository match the provided lightweight specifications, we invoke an SMT solver to ***solve the search***. Our current implementation uses Z3.

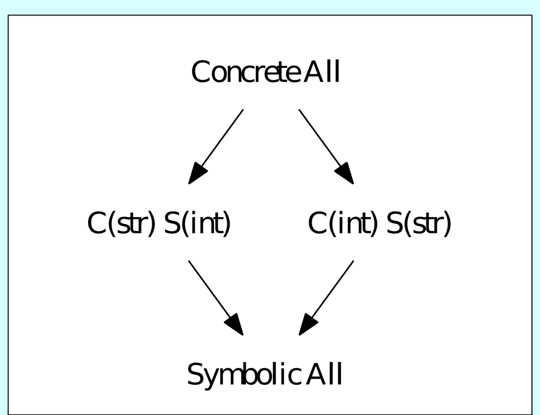
Encoding. Offline, a repository of programs is encoded as constraints. The level of granularity for encoding must balance the cost of the search (a level too fine could result in a constraint system that cannot be resolved) with the precision of matches (a level too coarse could return too many matches).



Module	Type	Constraint Def
1	equality	$out1 = i$
link(1,2)	equality	$in2 = out1$
2	inclusion	$(contains(in2,r) \wedge substr(field(r),c)) \rightarrow contains(out2,r)$
	exclusion	$contains(out2,r) \rightarrow contains(in2,r)$
	order	$\forall r_1,r_2((contains(out2,r_1) \wedge contains(out2,r_2) \wedge (\exists i,j(record(out2,i)=r_1 \wedge record(out2,j)=r_2 \wedge i<j)) \rightarrow (\exists k,l(k<l \wedge record(in2,k)=r_1 \wedge record(in2,l)=r_2)))$
link(2,3)	equality	$in3 = out2$
3	inclusion	$\forall i(0 \leq i < n) \rightarrow record(in3,i) = record(out3,i))$
	exclusion	$contains(out3,r) \rightarrow contains(in3,r)$
	order	$\forall i(0 \leq i < size(out3) \rightarrow record(in3,i) = record(out3,i))$
link(3,4)	equality	$in4 = out3$
4	equality	$in4 = 0$

Refinement. If the specifications or encoded program constraints are too weak, many matches may be returned; if they are too strong, the solver may not yield any results. In both cases, refinement is needed.





Abstraction Lattice

Concrete:

$$(contains(in,r) \wedge substr(field(r),\text{“tennis”})) \rightarrow contains(out,r)$$

Symbolic:

$$\exists s \mid (contains(in,r) \wedge substr(field(r),s)) \rightarrow contains(out,r)$$

This work was supported in part by the NSF Graduate Research Fellowship under CFDA#47.076 and NSF Award #0915526.