Hosted miniKanren: reconciling optimizing compilation and extensibility

Michael Ballantyne (Northeastern University)
Mitch Gamburg
Jason Hemann (Seton Hall University)

The usual tradeoff

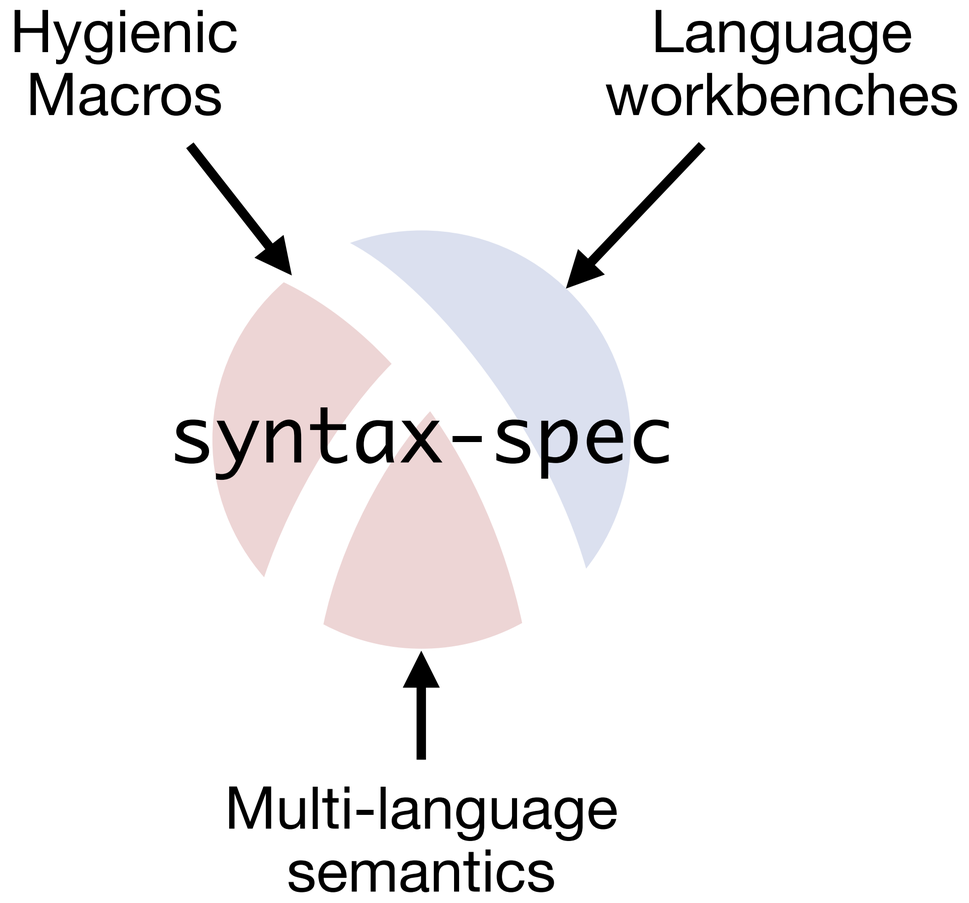
Shallow embeddings, conventional macros	• Extensibility • Easy interaction with host	core.logic, faster miniKanren, OCanren
Deep embeddings, language workbenches	• Optimizing compilation • Custom static semantics	Partial deduction, fair conjunction

A new way of implementing internal DSLs

- Optimizing compilation
- Custom static semantics

and

- Extensibility
- Easy interaction with host



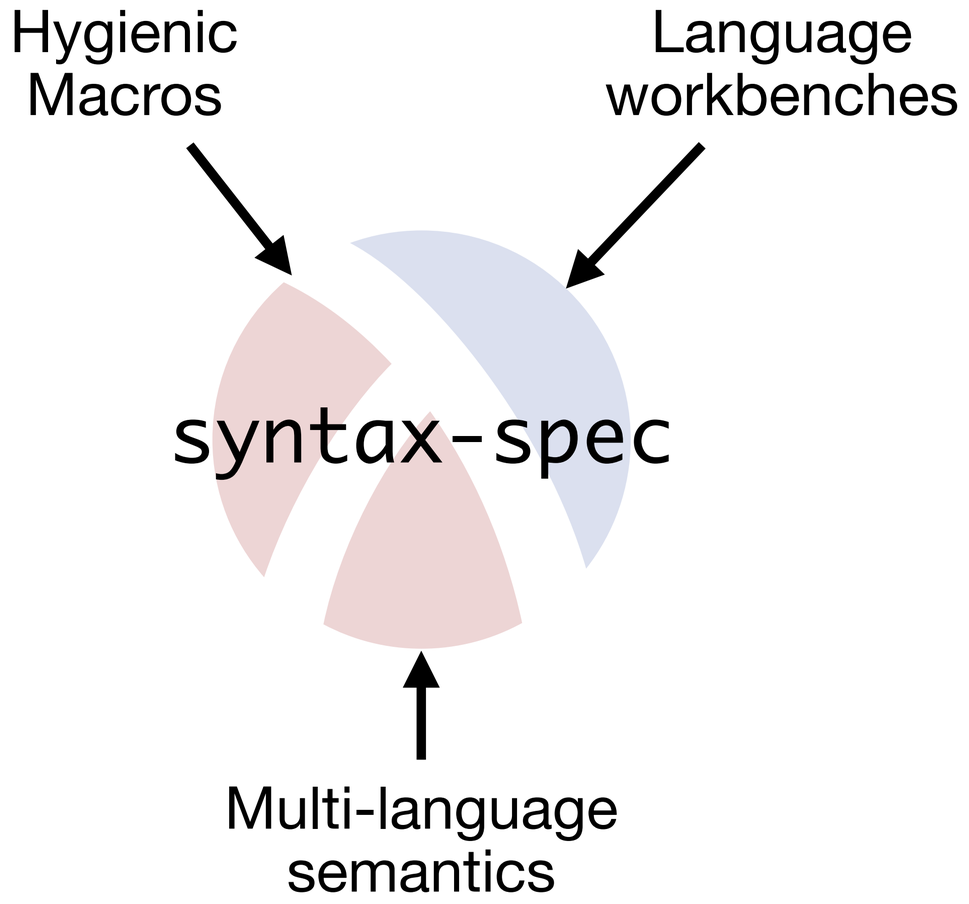
A new way of implementing internal DSLs

↪ A new implementation of miniKanren

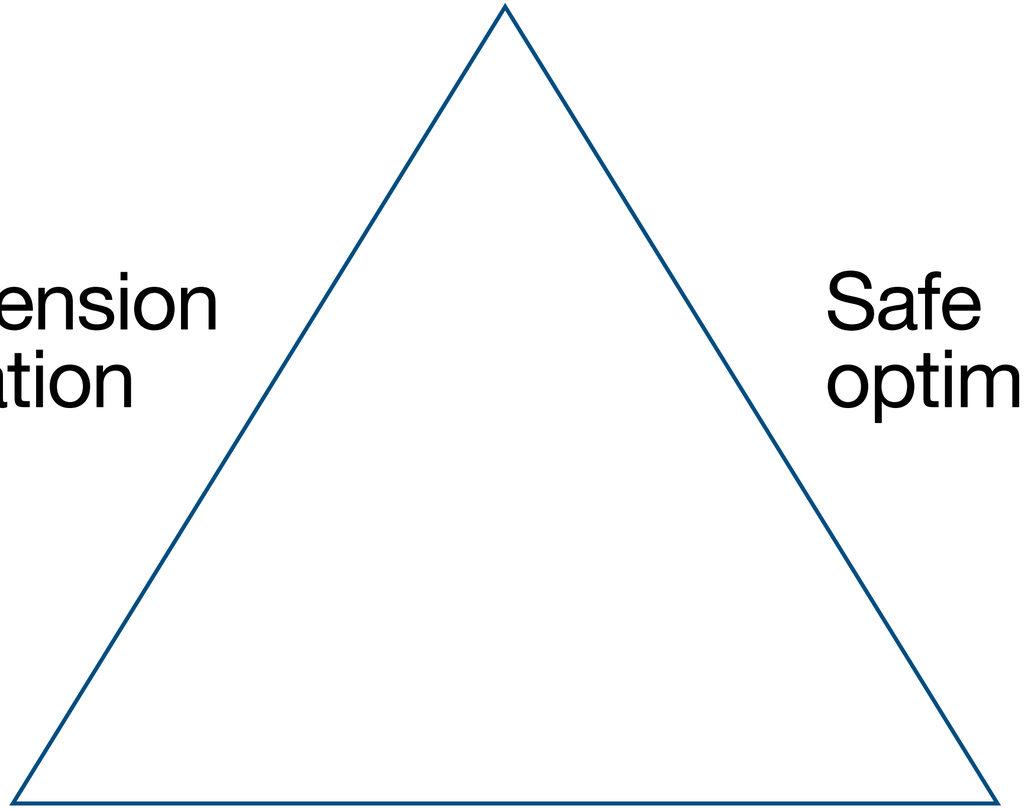
- Optimizing compilation
- Custom static semantics

and

- Extensibility
- Easy interaction with host



Optimizing compilation



Simpler extension implementation

Safe optimization

Extensions

Expressiveness

Multi-language

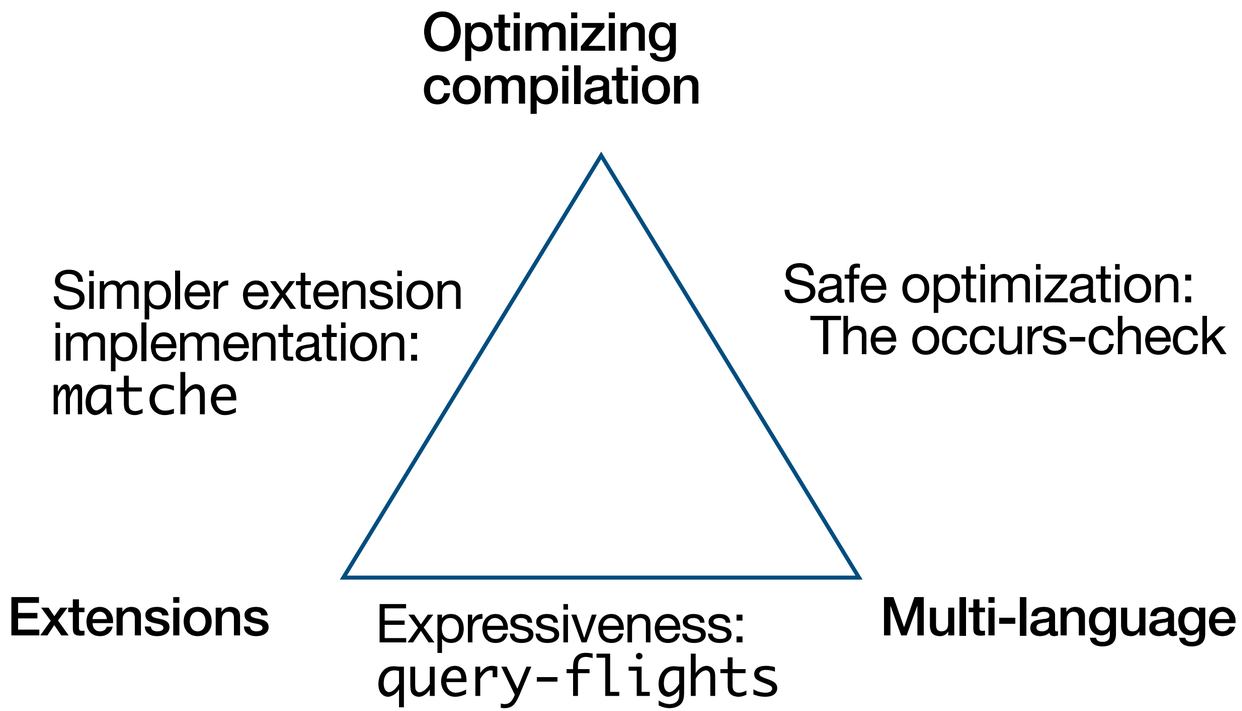
Demo

Benchmark	faster-mK	no opts	prop only	dead code	occurs check	specialization and overall
Occurs check						
`leo` 8000	209	1	1	1	52.25	69.67
appendo w/2 lists	327	0.99	1	1.01	81.75	109
Relational arithmetic						
`logo`	437	0.99	1	0.98	1.08	1.44
`four fours` of 256	77	0.95	0.97	0.94	1.01	1.26
`fact` x = 720	122	1.04	1.03	1.01	1.18	1.56
Relational interpreters						
one quine	962	0.99	0.96	0.97	1.01	1.01
9,900 `(I love you)`s	1378	0.96	0.95	0.98	1.01	1.04
append synthesis	252	1	1	0.98	1.33	1.81
dynamic and lexical	18	1	1.06	1	1.2	1.5
four thrines	683	0.98	1	1	1.03	1.06
countdown from 2 in λ-calc	64	0.97	1	1	6.4	7.11

Takeaways

By specifying:
- Grammar
- Binding rules
- Multi-language structure

We get:
- Syntax checking, IDE services
- Hygienic expansion to DSL core
- Static checking that works with separate compilation
- Boundaries that protect interactions



Try it out! And add an optimization?

“Compiled, Extensible, Multi-language DSLs"
github.com/michaelballantyne/hosted-minikanren
github.com/michaelballantyne/syntax-spec