PROJECTOR: an automatic logic program rewriting tool for better performance

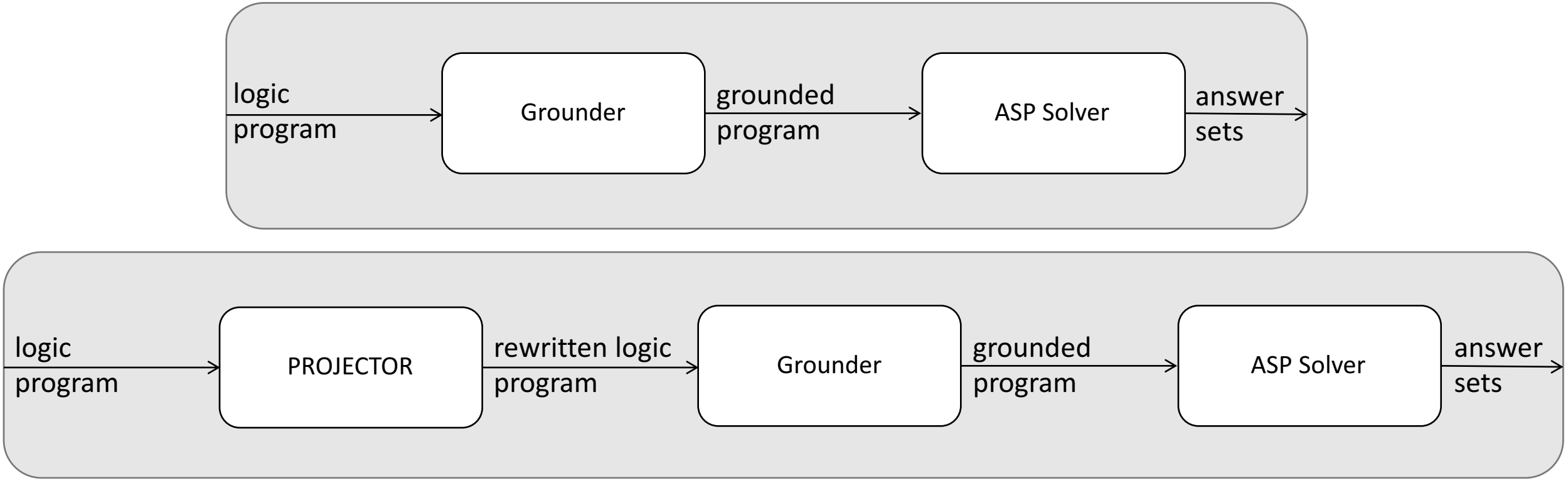
Nick Hippen & Yuliya Lierler

What is Answer Set Programming (ASP)?

- Constraint programming paradigm geared towards solving difficult combinatorial search problems
- Prolog-like syntax

Logic Rule	Meaning
$child(X,Y) \leftarrow parent(Y,X).$	*X is a child of Y if Y is a parent of X.*
$innocent(X) \leftarrow not\ guilty(X).$ Head ← Body	*X is innocent if I have no reason to believe that X is guilty*

ASP Solver Architecture



Grounding Logic Programs

Logic Program	Grounded Program	Intelligently Grounded Program
$parent(bob, ally).$ $parent(marry, john).$ $sibling(bob, marry).$ $cousin(X, Y) \leftarrow parent(P1, X),$ $\quad parent(P2, Y),$ $\quad sibling(P1, P2),$ $\quad X \neq Y.$	$parent(bob, ally).$ $parent(marry, john).$ $sibling(bob, marry).$ $cousin(john, marry) \leftarrow parent(ally, john),$ $\quad parent(bob, marry),$ $\quad sibling(ally, bob),$ $\quad john \neq marry.$ … $cousin(bob, bob) \leftarrow parent(bob, bob),$ $\quad parent(bob, bob),$ $\quad sibling(bob, bob),$ $\quad bob \neq bob.$ …	$parent(bob, ally).$ $parent(marry, john).$ $sibling(bob, marry).$ $cousin(ally, john) \leftarrow parent(bob, ally),$ $\quad parent(marry, john),$ $\quad sibling(bob, marry),$ $\quad ally \neq john.$

Improving Performance

Smaller grounding sizes often translate into faster solve times

Idea: Split a logic rule into multiple rules so that the number of variables present in each new rule is smaller than that of the original.

Projection
Two types: α and β

PROJECTOR Result: α

Logic Program	PROJECTOR: α-projection
$cousin(X,Y) \leftarrow parent(P1,X),$ $parent(P2,Y),$ $sibling(P1,P2),$ $X \neq Y.$	$p0(Y,P1) \leftarrow sibling(P1,P2), parent(P2,Y).$ $p1(Y,X) \leftarrow parent(P1,X), p0(Y,P1).$ $cousin(X,Y) \leftarrow X \neq Y, p1(Y,X).$

Nondeterministic behavior

Logic Program	PROJECTOR: α-projection Scenario #1	PROJECTOR: α-projection Scenario #2
$cousin(X,Y) \leftarrow parent(P1,X),$ $parent(P2,Y),$ $sibling(P1,P2),$ $X \neq Y.$	$p0(Y,P1) \leftarrow sibling(P1,P2), parent(P2,Y).$ $p1(Y,X) \leftarrow parent(P1,X), p0(Y,P1).$ $cousin(X,Y) \leftarrow X \neq Y, p1(Y,X).$	$p0(P2,X) \leftarrow sibling(P1,P2), parent(P1,X).$ $p1(Y,X) \leftarrow parent(P2,Y), p0(P2,X).$ $cousin(X,Y) \leftarrow X \neq Y, p1(Y,X).$

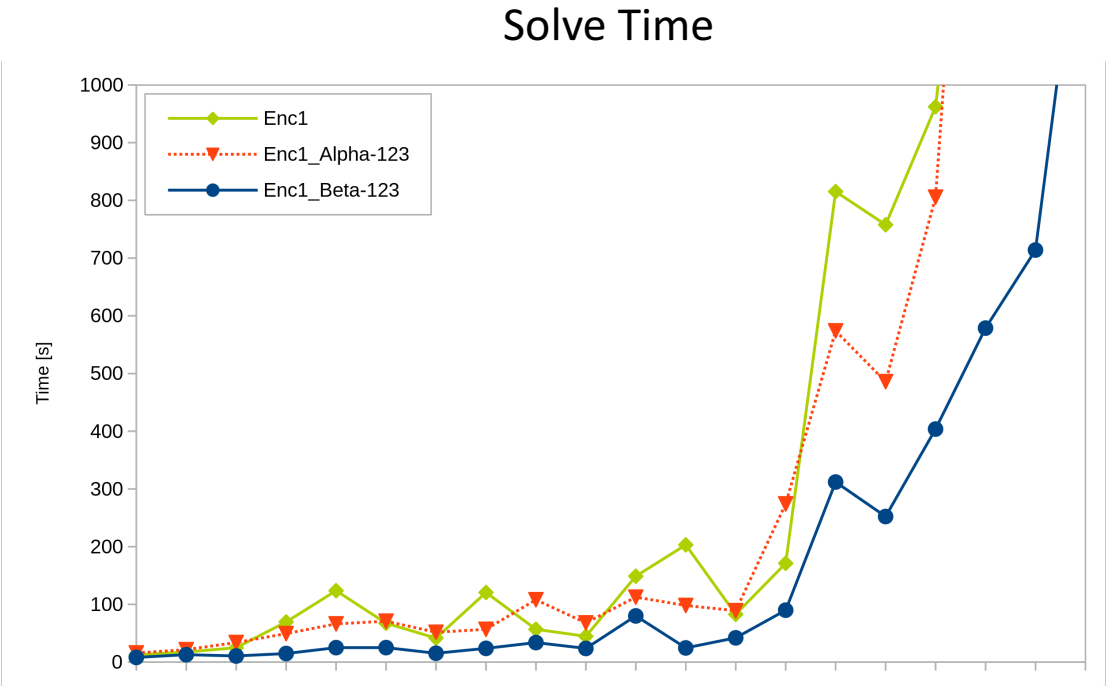
PROJECTOR Result: β

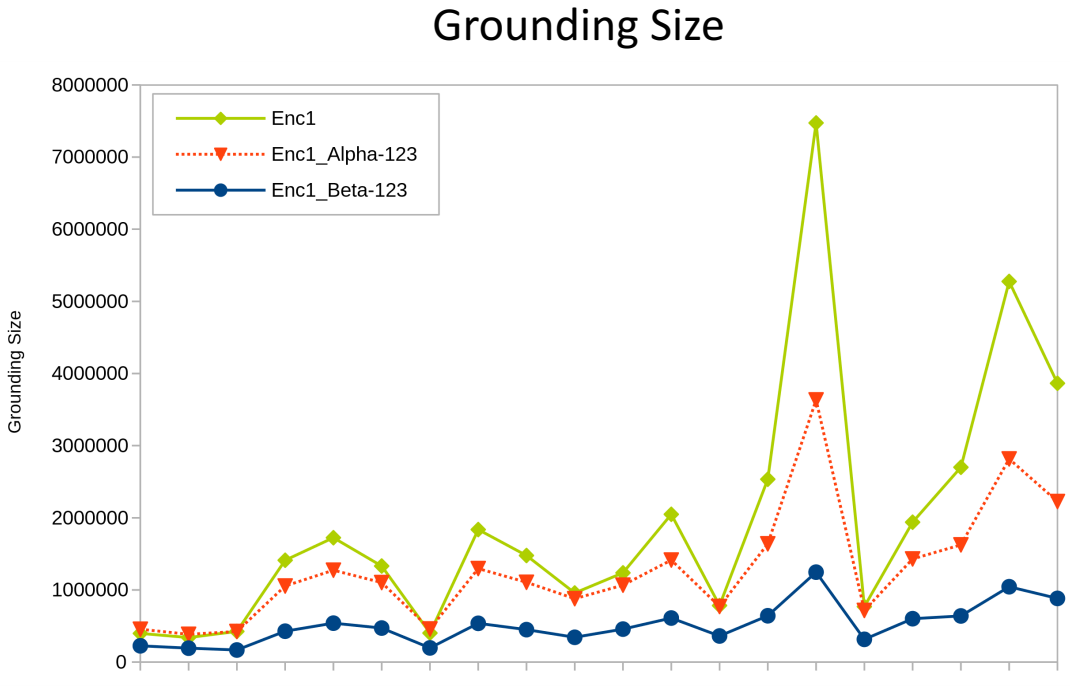
Logic Program	PROJECTOR: α-projection	PROJECTOR: β-projection
$male_{\downarrow}cousin(X,Y) \leftarrow parent(P1,X),$ $parent(P2,Y),$ $sibling(P1,P2),$ $X \neq Y$ $male(X).$	$p0(Y,P1) \leftarrow sibling(P1,P2), parent(P2,Y).$ $p1(Y,X) \leftarrow parent(P1,X), p0(Y,P1).$ $male_{\downarrow}cousin(X,Y) \leftarrow X \neq Y, male(X), p1(Y,X).$	$p0(Y,P1) \leftarrow sibling(P1,P2), parent(P2,Y).$ $p1(Y,X) \leftarrow parent(P1,X), p0(Y,P1), male(X).$ $male_{\downarrow}cousin(X,Y) \leftarrow X \neq Y, male(X), p1(Y,X).$

Experimental Analysis

- **ASPCCG**: ASP based natural language parser
  - 3 encodings of increasing levels of human optimization
    - Created by Matthew Buddenhagen, Yuliya Lierler & Peter Schuller
  - Enc1: No human optimization
  - Enc7: Moderate human optimization
  - Enc19: Notable human optimization

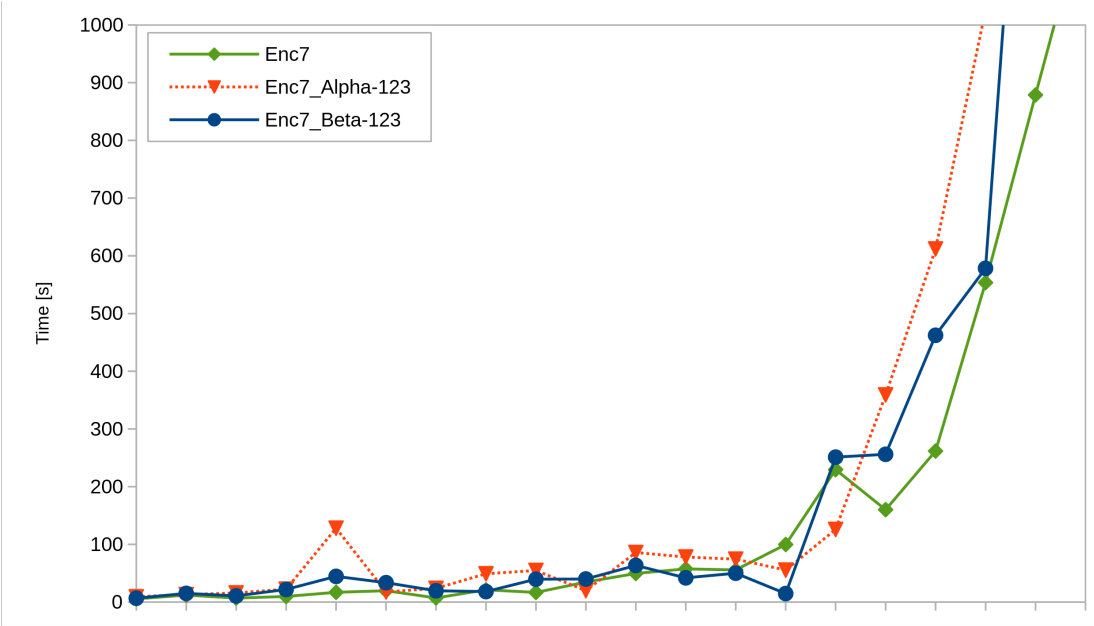
ASPCCG: Encoding 1



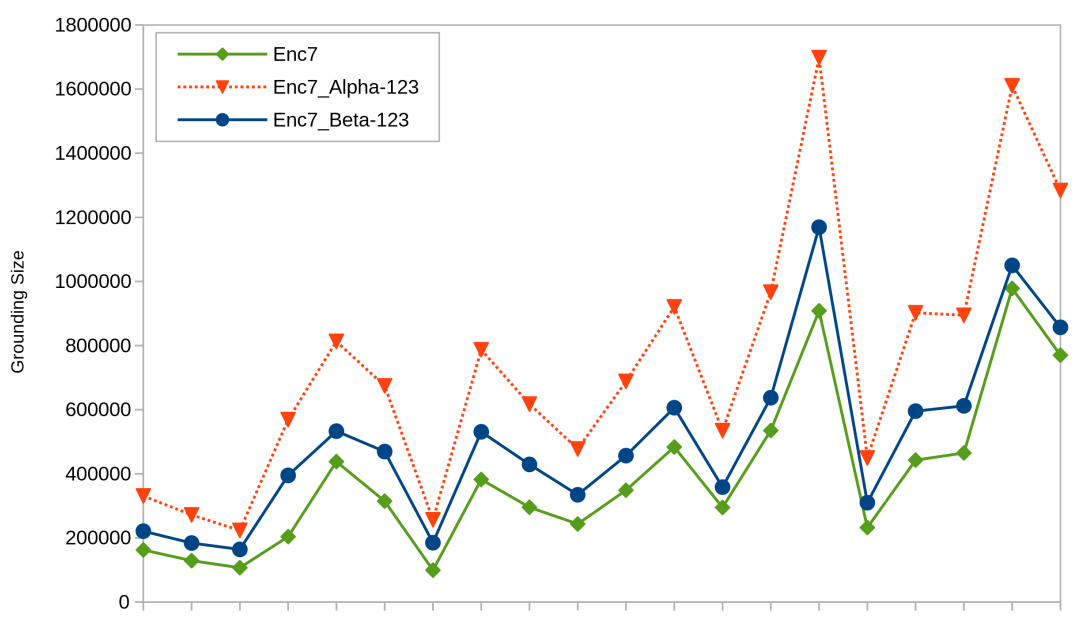


ASPCCG: Encoding 7

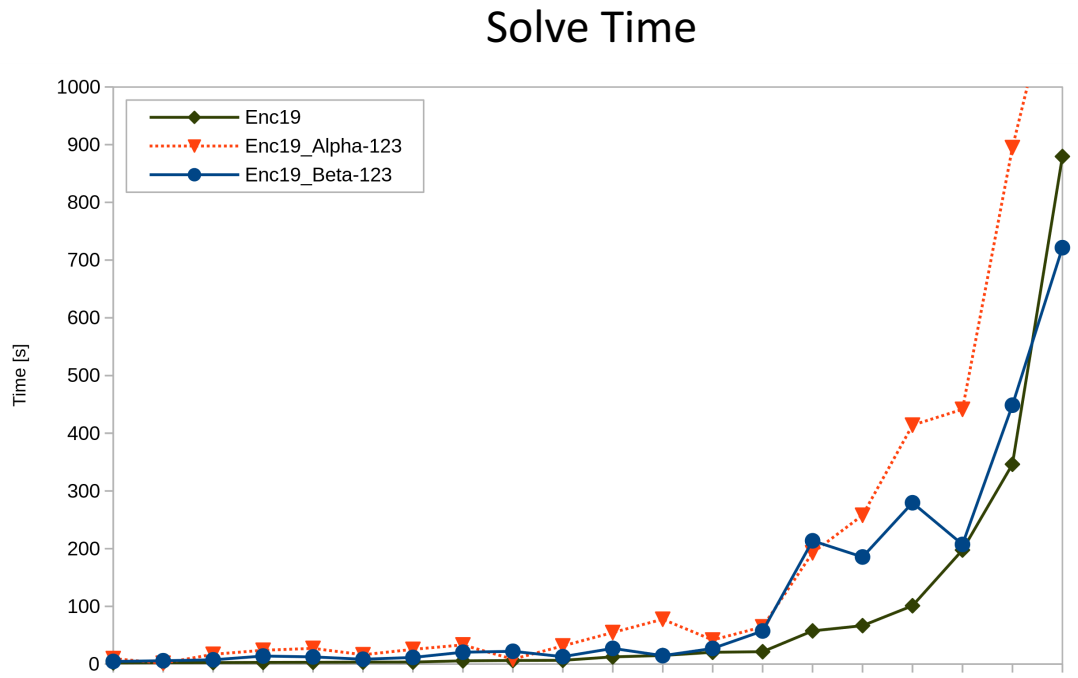
Solve Time

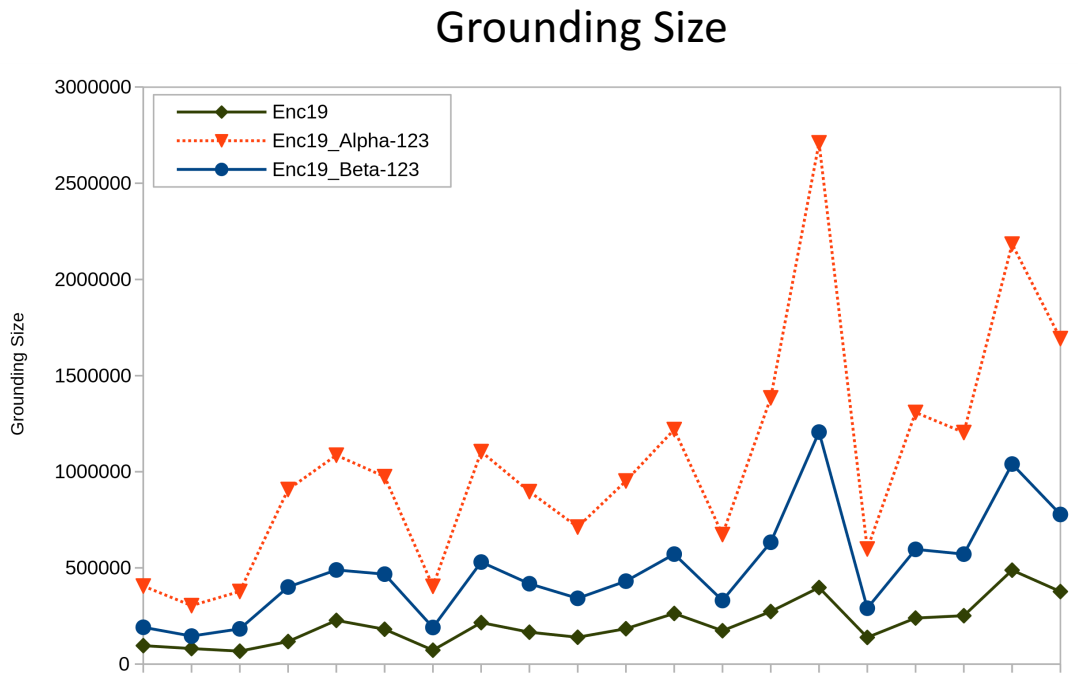


Grounding Size

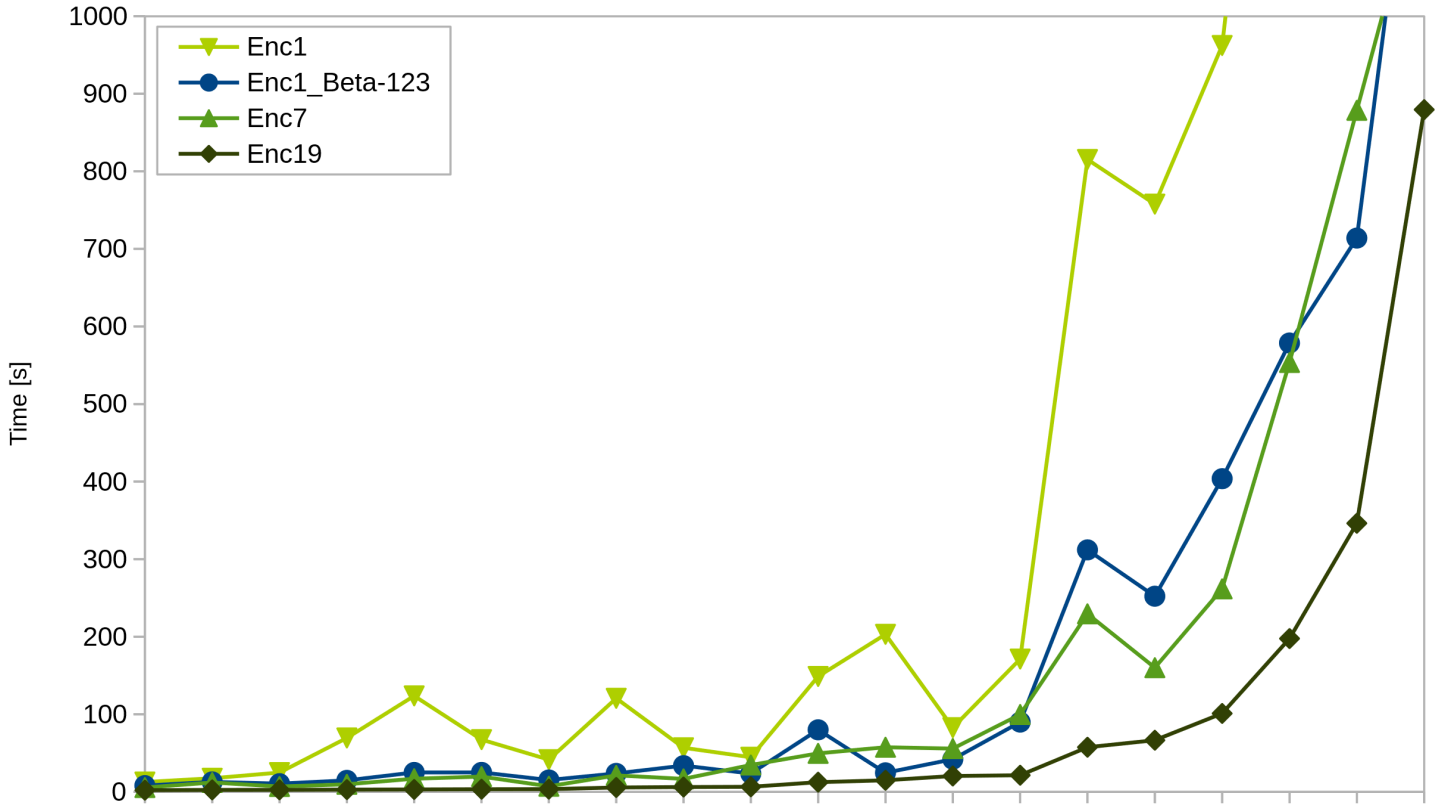


ASPCCG: Encoding 19





ASPCCG: Overall



Related, Current & Future Work

- Related work: lpopt (Bichler, Morak, Woltran, 2016)
- Paper will be submitted to Practical Aspects of Declarative Languages (PADL) 2019 this weekend
- System PROJECTOR available on the UNO NLPKR Lab website

Future Work

- Gather more benchmarks
- Grounding size prediction
- Improve language support

Acknowledgements

- Michael Dingess
- Brian Hodges
- Daniel Houston
- Roland Kaminski
- Liu Liu
- Dr. Mirek Truszczynski
- Stefan Woltran

Questions?