# Semantic Foundations of Equality Saturation

Dan Suciu<sup>1</sup>, Remy Wang<sup>2</sup>, <u>Yihong Zhang<sup>1</sup></u> <sup>1</sup> University of Washington <sup>2</sup> University of California, Los Angeles

Invented in 1970s and 2000s and repopularized in 2020.

- Invented in 1970s and 2000s and repopularized in 2020.
- Adopted in 50+ projects since then.

- Invented in 1970s and 2000s and repopularized in 2020.
- Adopted in 50+ projects since then.

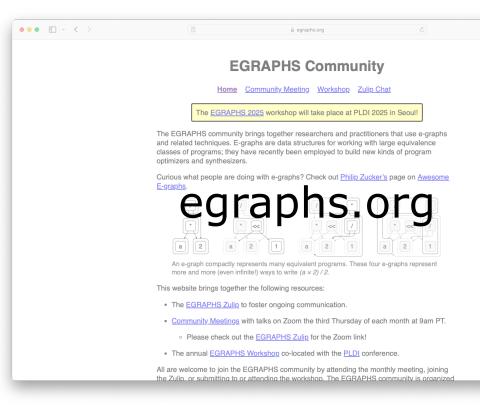
	Projects that use modern EqSat libraries		
Aa Project Name		i≣ Tags	■ Recognition
🚖 Isaria	Automatic Generation of Vectorizing Compilers for Customizable Digital Signal Processors	ASPLOS 2024	Best paper
🖕 Ruler	Rewrite Rule Inference Using Equality Saturation	OOPSLA 2021	Distinguished paper
🖕 Herbie	Automatically Improving Accuracy for Floating Point Expressions	PLDI 2015	Distinguished paper
🖕 MegaLibm	Implementation and Synthesis of Math Library Functions	POPL 2024	Distinguished paper
ROVER	Combining Power and Arithmetic Optimization via Datapath Rewriting	ARITH 2024 TCAD	
Diospyros	Vectorization for Digital Signal Processors via Equality Saturation	ASPLOS 2021	
Infinity Stream	Infinity Stream: Portable and Programmer-Friendly In-/Near-Memory Fusion	ASPLOS 2023	
SEER	SEER: Super-Optimization Explorer for HLS using E-graph Rewriting with MLIR	ASPLOS 2024	
Felix	Felix: Optimizing Tensor Programs with Gradient Descent	ASPLOS 2024	
Chassis	Target-Aware Implementation of Real Expressions	ASPLOS 2025	
Symbolics.jl	High-performance symbolic-numerics via multiple dispatch	CCA 2021	
MetaEmu	N vare	CCS 2022	
LIAR	L Ility Saturation	CGO 20	
⊽SD	COUNT 68	CGO 20	
DialEgg		CGO 20	
Zhan et al.	a Litting of Deep learning Operator	CGO 20	F
CvxLean	ransforming Optimization Problems into Disciplined Convex Programming Form	CICM 20	
wasm-evasion	WebAssembly diversification for malware evasion	COSE 2	
COUNT	68		

- Invented in 1970s and 2000s and repopularized in 2020.
- Adopted in 50+ projects since then.
- Thriving community
  - Zulip online chat
  - Monthly community meetings
  - Annual workshops

	Projects that use modern EqSat libraries		
Aa Project Name		i≣ Tags	■ Recognition
🚖 Isaria	Automatic Generation of Vectorizing Compilers for Customizable Digital Signal Processors	ASPLOS 2024	Best paper
🖕 Ruler	Rewrite Rule Inference Using Equality Saturation	OOPSLA 2021	Distinguished paper
🖕 Herbie	Automatically Improving Accuracy for Floating Point Expressions	PLDI 2015	Distinguished paper
🖕 MegaLibm	Implementation and Synthesis of Math Library Functions	POPL 2024	Distinguished paper
ROVER	Combining Power and Arithmetic Optimization via Datapath Rewriting	ARITH 2024 TCAD	
Diospyros	Vectorization for Digital Signal Processors via Equality Saturation	ASPLOS 2021	
Infinity Stream	Infinity Stream: Portable and Programmer-Friendly In-/Near-Memory Fusion	ASPLOS 2023	
SEER	SEER: Super-Optimization Explorer for HLS using E-graph Rewriting with MLIR	ASPLOS 2024	
Felix	Felix: Optimizing Tensor Programs with Gradient Descent	ASPLOS 2024	
Chassis	Target-Aware Implementation of Real Expressions	ASPLOS 2025	
Symbolics.jl	High-performance symbolic-numerics via multiple dispatch	CCA 2021	
MetaEmu	N vare	CCS 2022	
LIAR	L Ility Saturation	CGO 20	
⊽SD	COUNT 68	CGO 20	
DialEgg		CGO 20	
Zhan et al.	a Litting of Deep learning Operator	CGO 20	F
CvxLean	ransforming Optimization Problems into Disciplined Convex Programming Form	CICM 20	
wasm-evasion	WebAssembly diversification for malware evasion	COSE 2	
COUNT	68		

- Invented in 1970s and 2000s and repopularized in 2020.
- Adopted in 50+ projects since then.
- Thriving community
  - Zulip online chat
  - Monthly community meetings
  - Annual workshops

Aa Project Name	Paper title		i≣ Tags	Recognitio
🖕 Isaria	Automatic Generation of Vectorizing Compilers for Customizable Dig	aital Signal Processors	ASPLOS 2024	Best paper
숨 Ruler	Rewrite Rule Inference Using Equality Saturation		OOPSLA 2021	Distinguished paper
🖕 Herbie	Automatically Improving Accuracy for Floating Point Expressions		PLDI 2015	Distinguished
🖕 MegaLibm	Implementation and Synthesis of Math Library Functions		POPL 2024	Distinguished paper
ROVER	Combining Power and Arithmetic Optimization via Datapath Rewritin	g	ARITH 2024 TCAD	
Diospyros	Vectorization for Digital Signal Processors via Equality Saturation		ASPLOS 2021	
Infinity Stream	Infinity Stream: Portable and Programmer-Friendly In-/Near-Memory	Fusion	ASPLOS 2023	
SEER	SEER: Super-Optimization Explorer for HLS using E-graph Rewriting	with MLIR	ASPLOS 2024	
Felix	Felix: Optimizing Tensor Programs with Gradient Descent		ASPLOS 2024	
Chassis	Target-Aware Implementation of Real Expressions		ASPLOS 2025	
Symbolics.jl	High-performance symbolic-numerics via multiple dispatch		CCA 2021	
MetaEmu		vare	CCS 2022	···· ··· ··· · ·· -
LIAR		ility Saturation	CGO 20	
⊽SD	COUNT 68		CGO 20	
DialEgg			CGO 20	
Zhan et al.	d Lifting of Deep learning Operator		CGO 20	<b>F</b>
CvxLean	ransforming Optimization Problems into Disciplined Convex Progra	mming Form	CICM 2C	
wasm-evasion	WebAssembly diversification for malware evasion		COSE 2(	



Û	+	G

- Invented in 1970s and 2000s and repopularized in 2020.
- Adopted in 50+ projects since then.
- Thriving community
  - Zulip online chat
  - Monthly community meetings
  - Annual workshops

	Projects that use modern EqSat libraries			
Aa Project Name	🚍 Paper title		$\equiv$ Tags	■ Recognition
🖕 Isaria	Automatic Generation of Vectorizing Compilers for Customizable Dig	ital Signal Processors	ASPLOS 2024	Best paper
🚖 Ruler	Rewrite Rule Inference Using Equality Saturation		OOPSLA 2021	Distinguished paper
쑺 Herbie	Automatically Improving Accuracy for Floating Point Expressions		PLDI 2015	Distinguished paper
🖕 MegaLibm	Implementation and Synthesis of Math Library Functions		POPL 2024	Distinguished paper
ROVER	Combining Power and Arithmetic Optimization via Datapath Rewriting	1	ARITH 2024 TCAD	
Diospyros	Vectorization for Digital Signal Processors via Equality Saturation		ASPLOS 2021	
Infinity Stream	Infinity Stream: Portable and Programmer-Friendly In-/Near-Memory	Fusion	ASPLOS 2023	
SEER	SEER: Super-Optimization Explorer for HLS using E-graph Rewriting	with MLIR	ASPLOS 2024	
Felix	Felix: Optimizing Tensor Programs with Gradient Descent		ASPLOS 2024	
Chassis	Target-Aware Implementation of Real Expressions		ASPLOS 2025	
Symbolics.jl	High-performance symbolic-numerics via multiple dispatch		CCA 2021	
MetaEmu		vare	CCS 2022	
LIAR		lity Saturation	CGO 20	
⊽SD	COUNT 68		CGO 20	
DialEgg			CGO 20	
Zhan et al.	d Litting of Deep learning Operator		CGO 20	
CvxLean	ransforming Optimization Problems into Disciplined Convex Program	nming Form	CICM 20	
wasm-evasion	WebAssembly diversification for malware evasion		COSE 2(	
COUNT	<u>88</u>			

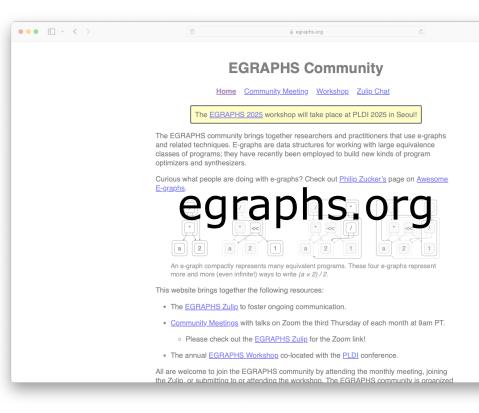
• • • • • • < >		🔒 egraphs.org	Ċ
	EG	RAPHS Communi	ty
	Home	Community Meeting Workshop Zulip	o Chat
	The EGRAPHS	2025 workshop will take place at PLDI	2025 in Seoul!
	and related techniques. E-gi	brings together researchers and practiti raphs are data structures for working wi ave recently been employed to build ner	th large equivalence
	E-graphs	ing with e-graphs? Check out Philip Zuc	<u>ker's</u> page on <u>Awesom</u>
		resents many equivalent programs. These f hite!) ways to write <i>(a × 2) / 2</i> .	our e-graphs represent
	This website brings together	r the following resources:	
	<ul> <li>The <u>EGRAPHS Zulip</u> to</li> </ul>	foster ongoing communication.	
	<u>Community Meetings</u> w	ith talks on Zoom the third Thursday of	each month at 9am PT
	<ul> <li>Please check out t</li> </ul>	he <u>EGRAPHS Zulip</u> for the Zoom link!	
	<ul> <li>The annual <u>EGRAPHS</u></li> </ul>	Workshop co-located with the PLDI cor	nference.
		EGRAPHS community by attending the attending the workshop. The EGRAPH	

)	Z	ulip - egraphs			
e-graphs	③ Recent conversion	ations Overview of ongoing conversa	. Q	Search ⑦	۵ 🎆
▼ VIEWS	Standard view	✓ □ Include DMs □ Unread	O Pa	articipated	
<ul> <li>Recent conversations</li> <li>Inbox</li> </ul>	Filter topics (t)				
Combined feed Mentions	Channel	Торіс	Ŀ,	Participants	Time 🔻
Reactions Starred messages		ground vs non-ground theorem proving			38 minute ago
Drafts	🔇 general	Zulip updates		8	4 days ago
<ul> <li>DIRECT MESSAGES</li> <li>Max Willsey</li> </ul>	📀 egg/general	Match arbitrary number of children			4 days ago
<ul> <li>Max Willsey, Remy Wang</li> <li>Rupanshu Soi</li> </ul>	🔇 egg/egglog	`:merge` function survey		**** *****	4 days ago
• anjali pal, Gus Smith, Oliv	er F	How to use egraph-serielize for		<b>法</b> 律	5 days ago
Contraction B Contractor	DNS.ZU	How to use egraph-serialize for	JE	<b>.</b> CO	6 dars ag
• Welcome Bot 🚵	S topic/applications	Rewrite rules for game state instead of expressions		8	7 days ago
CHANNELS Active	S general	extraction gym algorithms misses an optimization		<b></b>	11 days ag
egg/egglog egg/general	S general	Minimizing Egg explanations		2002	13 days ag
<ul> <li>general</li> <li>jobs</li> </ul>	S egg/egglog	✓ Get the Substitution in E- matching		⊇阒蘂攀	18 days ag
<ul> <li>topic/applications</li> <li>topic/implementation</li> </ul>	🚱 general	EGRAPHS Meeting 2025-02-20: CC Lemma			18 days a
6 topic/theory	🚱 egg/egglog	✓ Error calculating the power of rational numbers		■ ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ● ●	19 days ag
# egg/proof-checking	S topic/applications	Where are the rewrite rules?			20 days a
#E BROWSE 3 MORE CHANNEL	S Compose message		Start new	conversation New	direct messa

Û	+	G

- Invented in 1970s and 2000s and repopularized in 2020.
- Adopted in 50+ projects since then.
- Thriving community
  - Zulip online chat
  - Monthly community meetings
  - Annual workshops

	Projects that use modern EqSat libraries			
Aa Project Name	Paper title		i≣ Tags	■ Recognition
🖕 Isaria	Automatic Generation of Vectorizing Compilers for Customizable Digit	tal Signal Processors	ASPLOS 2024	Best paper
숨 Ruler	Rewrite Rule Inference Using Equality Saturation		OOPSLA 2021	Distinguished paper
🖕 Herbie	Automatically Improving Accuracy for Floating Point Expressions		PLDI 2015	Distinguished paper
🖕 MegaLibm	Implementation and Synthesis of Math Library Functions		POPL 2024	Distinguished paper
ROVER	Combining Power and Arithmetic Optimization via Datapath Rewriting		ARITH 2024 TCAD	
Diospyros	Vectorization for Digital Signal Processors via Equality Saturation		ASPLOS 2021	
Infinity Stream	Infinity Stream: Portable and Programmer-Friendly In-/Near-Memory I	Fusion	ASPLOS 2023	
SEER	SEER: Super-Optimization Explorer for HLS using E-graph Rewriting w	vith MLIR	ASPLOS 2024	
Felix	Felix: Optimizing Tensor Programs with Gradient Descent		ASPLOS 2024	
Chassis	Target-Aware Implementation of Real Expressions		ASPLOS 2025	
Symbolics.jl	High-performance symbolic-numerics via multiple dispatch		CCA 2021	
MetaEmu		vare	CCS 2022	
LIAR		lity Saturation	CGO 20	
∇SD	COUNT 68		CGO 20	
DialEgg			CGO 20	
Zhan et al.	d Litting of Deep learning Operator		CGO 20	F
CvxLean	ransforming Optimization Problems into Disciplined Convex Program	nming Form	CICM 20	
wasm-evasion	WebAssembly diversification for malware evasion		COSE 2(	
COUNT	38			



		Zulip	- egraphs			
	e-graphs	④ Recent conversati	ons Overview of ongoing conversa.	Q	Search	0 🌣 🎆
	VIEWS	Standard view ~	Include DMs Unread	D Pa	rticipated	
	<ul> <li>④ Recent conversations</li> <li>☑ Inbox</li> </ul>	Filter topics (t)				
	Combined feed Mentions	Channel	Торіс	Ŀ°	Participants	Time 👻
l	<ul> <li>Reactions</li> <li>Starred messages</li> <li>4</li> </ul>	🔇 topic/theory	ground vs non-ground theorem proving		<b></b>	38 minutes ago
	Drafts	🚯 general	Zulip updates		8	4 days ago
l	DIRECT MESSAGES     Max Willsey	📀 egg/general	Match arbitrary number of children		2	4 days ago
	<ul> <li>Max Willsey, Remy Wang</li> <li>Rupanshu Soi</li> </ul>	🔇 egg/egglog	`:merge` function survey		atts tooott troott	4 days ago
	• anjali pal, Gus Smith, Oliver F	S egg/egglog	How to use egraph-serialize for extractions on How to tread a tew base type of union?	<b>h</b>		5 days ago
t	Subjection B 🚺 P I I	<b>5</b> .2u	How to read a new bale type / union?	่วเ		6 dars ago
l	💿 Welcome Bot 눮	🔇 topic/applications	Rewrite rules for game state instead of expressions		8	7 days ago
l	CHANNELS Active	S general	extraction gym algorithms misses an optimization		<b></b>	11 days ag
	6 egg/egglog 6 egg/general	S general	Minimizing Egg explanations			13 days ag
l	<ul> <li>general</li> <li>iobs</li> </ul>	🔇 egg/egglog	✓ Get the Substitution in E- matching		■●靈働	18 days ag
	<ul> <li>topic/applications</li> <li>topic/implementation</li> </ul>	S general	EGRAPHS Meeting 2025-02-20: CC Lemma			18 days ag
1	S topic/theory	S egg/egglog	✓ Error calculating the power of rational numbers		2000 2000	19 days ag
	<pre># egg/proof-checking</pre>	S topic/applications	Where are the rewrite rules?			20 days ag
:	#E BROWSE 3 MORE CHANNELS	Compose message		Start new	conversation N	ew direct messa

Mon 16 - Fri 20 June 2025 Seoul, South Korea	
Attending - Tracks - Organization - Q Search Series -	
PLDI 2025 (series) / EGRAPHS 2025 (series) / EGRAPHS 2025	
About Call for Papers	Important Dates
Research in the EGRAPHS Community has recently exploded in both quantity and diversity. The data structure that powers SMT solvers is now seeing use in synthesis, optimization, and verification via equality saturation and related techniques. It addition to recent advances in the core data structure and techniques, researchers and practitioners are applying e-graphs to domains such as compilers, floating point accuracy, test generation, computational fabrication,	Thu 17 Apr 2025 Submission Deadlin
automatic vectorization, deep learning compute graphs, symbolic computation, and more. The fourth EGRAPHS workshop will bring together those working on and with e-graphs, providing a collaborative venue to share work that advances e-graphs as a broadly applicable technique in programming languages or other fields of computing. The program will contain a mix of invited speakers and work-in-progress talks. The symposium seeks papers on a diverse range of topics including (but not limited to):	Organizing Com
<ul> <li>e-graphs as data structures and their related algorithms</li> <li>equality saturation and other e-graph based rewriting approaches</li> <li>applications of e-graphs and/or equality saturation, whether in programming languages or other fields</li> <li>toois/irrameworks that facilitate the use of e-graphs and associated techniques</li> <li>investigations into the human-facing aspects using e-graph-based toolkits including error reporting, debugging, and visualization</li> <li>other frameworks for the ptimizing/analyzing programs in an equational manner</li> </ul>	Program Comm Samue Imperi United K
Accepted submissions will not be placed on the ACM DL, so we allow and encourage in-progress or already published relevant work to be presented.	Chris F F5

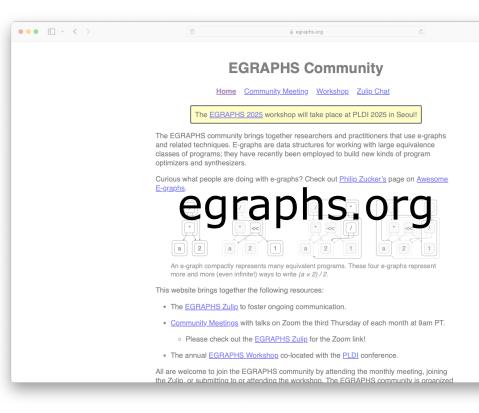
Û	+	6
		_

Sign in Sign up	
Sign in Sign up	
ITC-12h	
3	
ard	
ege London, UK / Intel	

- Invented in 1970s and 2000s and repopularized in 2020.
- Adopted in 50+ projects since then.
- Thriving community
  - Zulip online chat
  - Monthly community meetings
  - Annual workshops

Database theory community can help!

	Projects that use modern EqSat libraries			
Aa Project Name	Paper title		i≣ Tags	■ Recognition
🖕 Isaria	Automatic Generation of Vectorizing Compilers for Customizable Digit	tal Signal Processors	ASPLOS 2024	Best paper
숨 Ruler	Rewrite Rule Inference Using Equality Saturation		OOPSLA 2021	Distinguished paper
🖕 Herbie	Automatically Improving Accuracy for Floating Point Expressions		PLDI 2015	Distinguished paper
🖕 MegaLibm	Implementation and Synthesis of Math Library Functions		POPL 2024	Distinguished paper
ROVER	Combining Power and Arithmetic Optimization via Datapath Rewriting		ARITH 2024 TCAD	
Diospyros	Vectorization for Digital Signal Processors via Equality Saturation		ASPLOS 2021	
Infinity Stream	Infinity Stream: Portable and Programmer-Friendly In-/Near-Memory I	Fusion	ASPLOS 2023	
SEER	SEER: Super-Optimization Explorer for HLS using E-graph Rewriting w	vith MLIR	ASPLOS 2024	
Felix	Felix: Optimizing Tensor Programs with Gradient Descent		ASPLOS 2024	
Chassis	Target-Aware Implementation of Real Expressions		ASPLOS 2025	
Symbolics.jl	High-performance symbolic-numerics via multiple dispatch		CCA 2021	
MetaEmu		vare	CCS 2022	
LIAR		lity Saturation	CGO 20	
∇SD	COUNT 68		CGO 20	
DialEgg			CGO 20	
Zhan et al.	d Litting of Deep learning Operator		CGO 20	F
CvxLean	ransforming Optimization Problems into Disciplined Convex Program	nming Form	CICM 20	
wasm-evasion	WebAssembly diversification for malware evasion		COSE 2(	
COUNT	38			



		Zulip	- egraphs			
	e-graphs	④ Recent conversati	ons Overview of ongoing conversa.	Q	Search	0 🌣 🎆
	VIEWS	Standard view ~	Include DMs Unread	O Pa	rticipated	
	<ul> <li>④ Recent conversations</li> <li>☑ Inbox</li> </ul>	Filter topics (t)				
	Combined feed Mentions	Channel	Торіс	Ŀ°	Participants	Time 👻
l	<ul> <li>Reactions</li> <li>Starred messages</li> <li>4</li> </ul>	🔇 topic/theory	ground vs non-ground theorem proving		<b></b>	38 minutes ago
	Drafts	🚯 general	Zulip updates		8	4 days ago
l	DIRECT MESSAGES     Max Willsey	📀 egg/general	Match arbitrary number of children		2	4 days ago
	<ul> <li>Max Willsey, Remy Wang</li> <li>Rupanshu Soi</li> </ul>	🚱 egg/egglog	`:merge` function survey		atts thoots thoots	4 days ago
	• anjali pal, Gus Smith, Oliver F	S egg/egglog	How to use egraph-serialize for extractions of How to tread a tew bale type of union?	<b>h</b>		5 days ago
t	Subjection B 🚺 P I I	<b>5</b> .2u	How tocreally a new balle type / union?	al		6 dars ago
l	💿 Welcome Bot 눮	🔇 topic/applications	Rewrite rules for game state instead of expressions		8	7 days ago
l	CHANNELS Active	S general	extraction gym algorithms misses an optimization		<b></b>	11 days ag
	6 egg/egglog 6 egg/general	S general	Minimizing Egg explanations			13 days ag
l	<ul> <li>general</li> <li># jobs</li> </ul>	egg/egglog	✓ Get the Substitution in E- matching		20 靈樂	18 days ag
	<ul> <li>topic/applications</li> <li>topic/implementation</li> </ul>	S general	EGRAPHS Meeting 2025-02-20: CC Lemma			18 days ag
1	S topic/theory	S egg/egglog	✓ Error calculating the power of rational numbers		<b>こ側</b> 總羅	19 days ag
	<pre># egg/proof-checking</pre>	S topic/applications	Where are the rewrite rules?			20 days ag
:	#E BROWSE 3 MORE CHANNELS	Compose message		Start new	conversation N	ew direct messa

Mon 16 - Fri 20 June 2025 Seoul, South Korea	
Attending - Tracks - Organization - Q Search Series -	
PLDI 2025 (series) / EGRAPHS 2025 (series) / EGRAPHS 2025	
About Call for Papers	Important Dates
Research in the EGRAPHS Community has recently exploded in both quantity and diversity. The data structure that powers SMT solvers is now seeing use in synthesis, optimization, and verification via equality saturation and related techniques. In addition to recent advances in the core data structure and techniques, researchers and practitioners are applying e-graphs to domains such as compilers, floating point accuracy, test generation, computational fabrication,	Thu 17 Apr 2025 Submission Deadlin
automatic vectorization, deep learning compute graphs, symbolic computation, and more. The fourth EGRAPHS workshop will bring together those working on and with e-graphs, providing a collaborative venue to share work that advances e-graphs as a broadly applicable technique in programming languages or other fields of computing. The program will contain a mix of invited speakers and work-in-progress talks. The symposium seeks papers on a diverse range of topics including (but not limited to):	Organizing Com
<ul> <li>e-graphs as data structures and their related algorithms</li> <li>equality saturation and other e-graph based rewriting approaches</li> <li>applications of e-graphs and/or equality saturation, whether in programming languages or other fields</li> <li>tools/rameworks that facilitate the use of e-graphs and associated techniques</li> <li>investigations into the human-facing aspects using e-graph-based toolkits including error reporting, debugging, and visualization</li> <li>other frameworks for optimizing/analyzing programs in an equational manner</li> </ul>	Program Comm
Accepted submissions will not be placed on the ACM DL, so we allow and encourage in-progress or already published	Chris F F5

Û	+	6
		_

Sign in Sign up	
Sign in Sign up	
ITC-12h	
3	
ard	
ege London, UK / Intel	

#### This paper

- Defines a rigorous semantics to Equality Saturation (EqSat).
- Studies EqSat in relationship to Term Rewriting and the Chase.
- Proves the undecidability of EqSat termination in three cases.

• Signature  $\Sigma := \{f_1, f_2, ...\}.$ 

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.
- Ground terms  $T_{\Sigma}$   $(:= T_{\Sigma}(\emptyset)).$

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.
- Ground terms  $T_{\Sigma}$   $(:= T_{\Sigma}(\emptyset)).$
- The (ground) word problem
  - Input:  $E = \{s_1 \approx t_1, \dots\}$  and  $u, v \in T_{\Sigma}$ .
  - Ask:  $u \stackrel{\cdot}{\approx}_E v$ .

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.
- Ground terms  $T_{\Sigma}$   $(:= T_{\Sigma}(\emptyset)).$
- The (ground) word problem
  - Input:  $E = \{s_1 \approx t_1, ...\}$  and  $u, v \in T_{\Sigma}$ .
  - Ask:  $u \stackrel{\cdot}{\approx}_E v$ .
- Undecidable in general.

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.
- Ground terms  $T_{\Sigma}$   $(:= T_{\Sigma}(\emptyset)).$
- The (ground) word problem
  - Input:  $E = \{s_1 \approx t_1, ...\}$  and  $u, v \in T_{\Sigma}$ .
  - Ask:  $u \stackrel{\cdot}{\approx}_E v$ .
- Undecidable in general.

• Term rewriting for word problem:

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.
- Ground terms  $T_{\Sigma}$   $(:= T_{\Sigma}(\emptyset)).$
- The (ground) word problem
  - Input:  $E = \{s_1 \approx t_1, ...\}$  and  $u, v \in T_{\Sigma}$ .
  - Ask:  $u \stackrel{\cdot}{\approx}_E v$ .
- Undecidable in general.

- Term rewriting for word problem:
  - Use a Term Rewriting System (TRS) *R* capturing axioms *E*.

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.
- Ground terms  $T_{\Sigma}$   $(:= T_{\Sigma}(\emptyset)).$
- The (ground) word problem
  - Input:  $E = \{s_1 \approx t_1, ...\}$  and  $u, v \in T_{\Sigma}$ .
  - Ask:  $u \stackrel{\cdot}{\approx}_E v$ .
- Undecidable in general.

- Term rewriting for word problem:
  - Use a Term Rewriting System (TRS) *R* capturing axioms *E*.
  - Apply  $\rightarrow_R$  to u and v and check if  $\exists w \, . \, u \rightarrow^*_R w \leftarrow^*_R v.$

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.
- Ground terms  $T_{\Sigma}$   $(:= T_{\Sigma}(\emptyset)).$
- The (ground) word problem
  - Input:  $E = \{s_1 \approx t_1, ...\}$  and  $u, v \in T_{\Sigma}$ .
  - Ask:  $u \stackrel{\cdot}{\approx}_E v$ .
- Undecidable in general.

- Term rewriting for word problem:
  - Use a Term Rewriting System (TRS) *R* capturing axioms *E*.
  - Apply  $\rightarrow_R$  to u and v and check if  $\exists w . u \rightarrow_R^* w \leftarrow_R^* v.$

Problem:  $(a^{-1} \cdot b^{-1})^{-1} \stackrel{?}{\approx} b \cdot a$ 

- Signature  $\Sigma := \{f_1, f_2, ...\}.$
- Patterns  $T_{\Sigma}(V)$  for set of vars V.
- Ground terms  $T_{\Sigma}$   $(:= T_{\Sigma}(\emptyset)).$
- The (ground) word problem
  - Input:  $E = \{s_1 \approx t_1, ...\}$  and  $u, v \in T_{\Sigma}$ .
  - Ask:  $u \stackrel{\cdot}{\approx}_E v$ .
- Undecidable in general.

- Term rewriting for word problem:
  - Use a Term Rewriting System (TRS) *R* capturing axioms *E*.
  - Apply  $\rightarrow_R$  to u and v and check if  $\exists w \, . \, u \rightarrow^*_R w \leftarrow^*_R v.$

Problem: 
$$(a^{-1} \cdot b^{-1})^{-1} \stackrel{?}{\approx} b \cdot a$$
  
Yes.  
 $(a^{-1} \cdot b^{-1})^{-1} \rightarrow (b^{-1})^{-1} \cdot (a^{-1})^{-1}$   
 $\rightarrow b \cdot (a^{-1})^{-1}$   
 $\rightarrow b \cdot a$ 

**Program Optimization** 

Program Optimization

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,

Program Optimization

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization

 $x \div x \to 1 \qquad (x \times y) \div z \to x \times (y \div z)$  $x \times 1 \to x \qquad x \times 2 \to x \ll 1$ 

. . .

**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization

 $x \div x \to 1 \qquad (x \times y) \div z \to x \times (y \div z)$  $x \times 1 \to x \qquad x \times 2 \to x \ll 1$ 

. . .

 $(a \times 2) \div 2$ 

**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization

 $x \div x \to 1 \qquad (x \times y) \div z \to x \times (y \div z)$  $x \times 1 \to x \qquad x \times 2 \to x \ll 1$ 

. . .

 $(a \times 2) \div 2 \longrightarrow a \times (2 \div 2)$ 

**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization  $x \div x \to 1$   $(x \times y) \div z \to x \times (y \div z)$ 

 $x \times 1 \to x \qquad \qquad x \times 2 \to x \ll 1$ 

. . .

 $(a \times 2) \div 2 \longrightarrow a \times (2 \div 2) \longrightarrow a \times 1$ 

**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization

 $x \div x \to 1 \qquad (x \times y) \div z \to x \times (y \div z)$  $x \times 1 \to x \qquad x \times 2 \to x \ll 1$ 

. . .

 $(a \times 2) \div 2 \longrightarrow a \times (2 \div 2) \longrightarrow a \times 1 \longrightarrow a$ 



**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization

 $x \div x \to 1 \qquad (x \times y) \div z \to x \times (y \div z)$  $x \times 1 \to x \qquad x \times 2 \to x \ll 1$ 

. . .

 $(a \times 2) \div 2 \longrightarrow a \times (2 \div 2) \longrightarrow a \times 1 \longrightarrow a$  $(a \ll 1)/2$ 



**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization

 $x \div x \to 1 \qquad (x \times y) \div z \to x \times (y \div z)$  $x \times 1 \to x \qquad x \times 2 \to x \ll 1$ 

. . .

 $(a \times 2) \div 2 \longrightarrow a \times (2 \div 2) \xrightarrow{} a \times 1 \xrightarrow{} a$  $(a \ll 1)/2 \xrightarrow{} ?$ 



**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function *C*,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization $x \div x \to 1$  $(x \times y) \div z \to x \times (y \div z)$  $x \times 1 \to x$  $x \times 2 \to x \ll 1$ 

. . .

 $(a \times 2) \div 2 \longrightarrow a \times (2 \div 2) \longrightarrow a \times 1 \longrightarrow a$ 

 $(a \ll 1)/2 \rightarrow ?$ 

Term rewriting is greedy!



**Program Optimization** 

- Input:
  - A set of axioms E,
  - A program *s*,
  - Cost function C,
- Output:
  - Optimized program

$$t = \arg\min_{t \in [s]_E} C(t)$$

where

$$[s]_E := \{t \mid t \in T_{\Sigma} \, . \, t \approx_E s\}.$$

Ruled-based program optimization  $x \div x \to 1$   $(x \times y) \div z \to x \times (y \div z)$  $x \times 2 \rightarrow x \ll 1$  $x \times 1 \rightarrow x$ 

. . .

 $(a \times 2) \div 2 \longrightarrow a \times (2 \div 2) \rightarrow a \times 1 \rightarrow a$ 

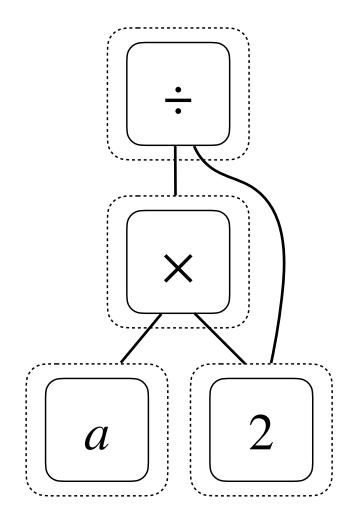
 $(a \ll 1)/2 \rightarrow ?$ 

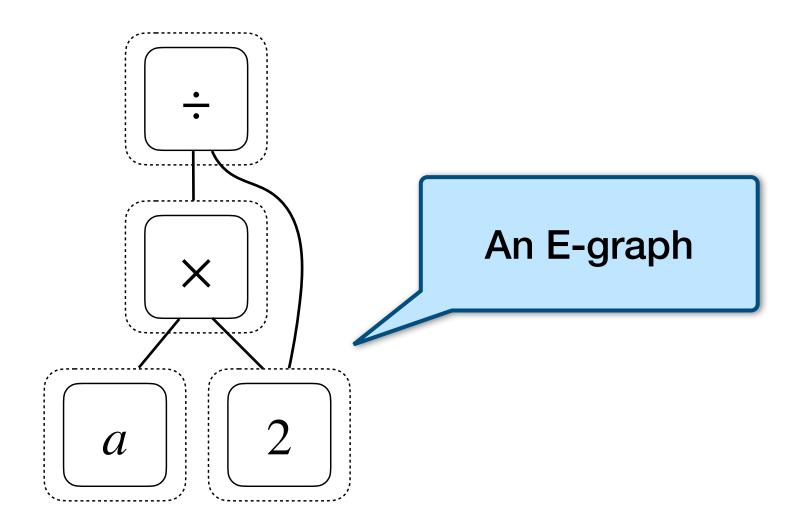
Term rewriting is greedy!

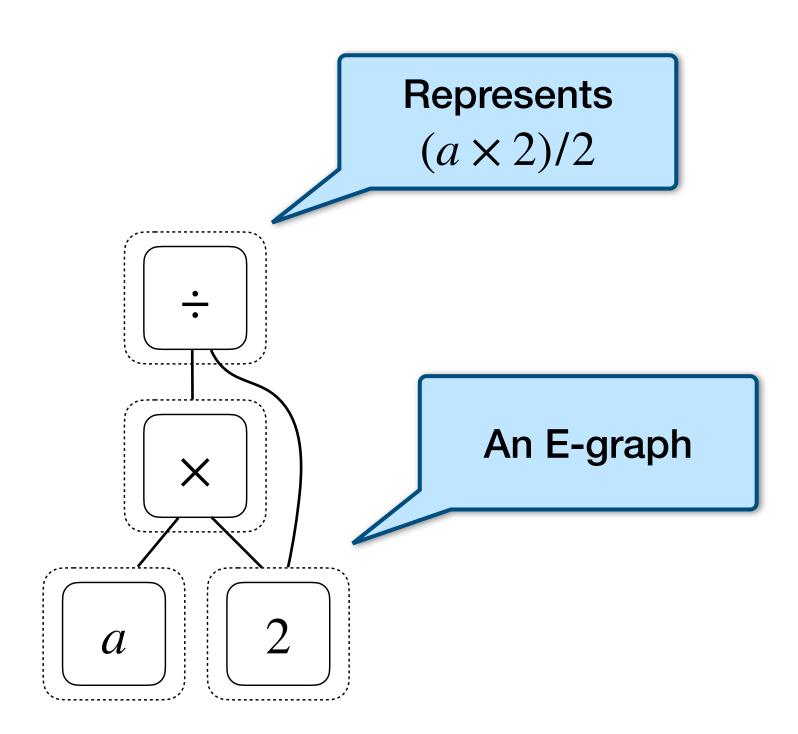
Equality Saturation is an algorithm to efficiently explore the program space defined by rules.

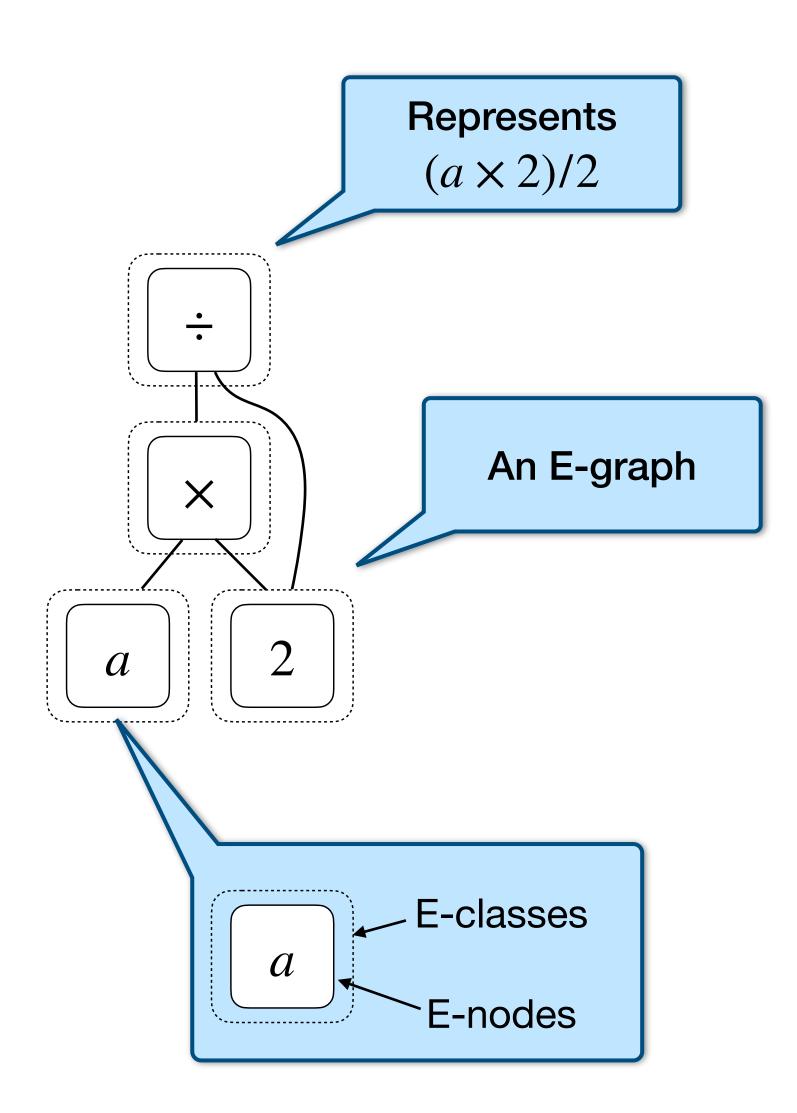


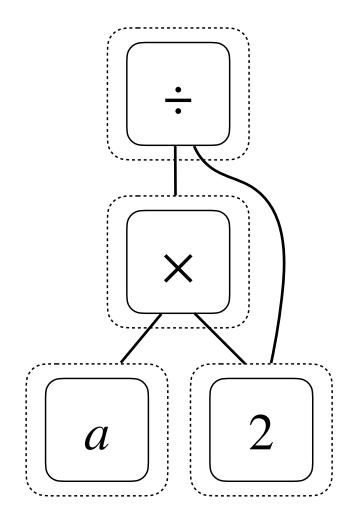
## **E-graphs and Equality Saturation**

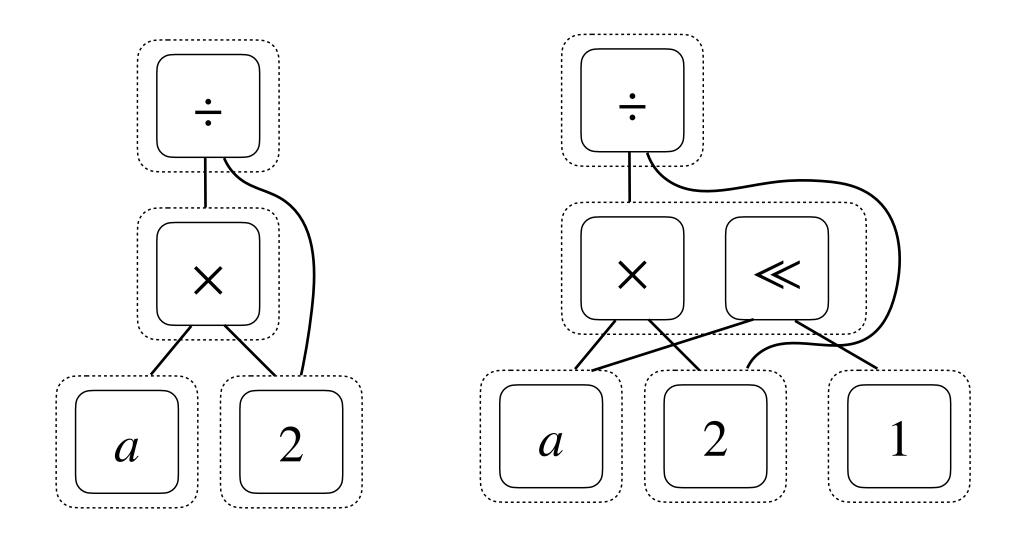




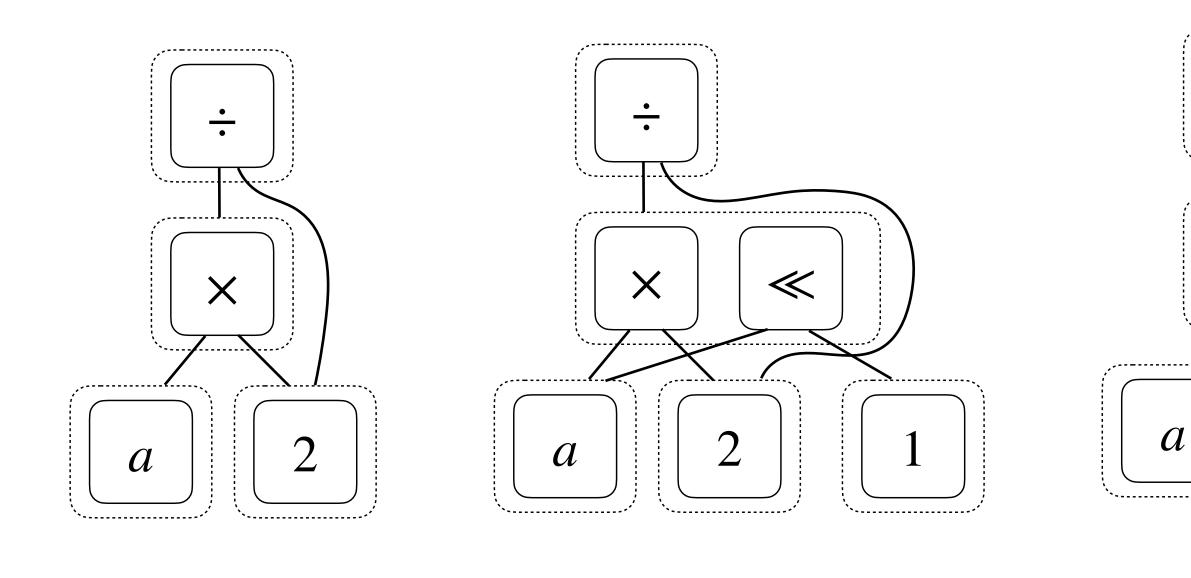




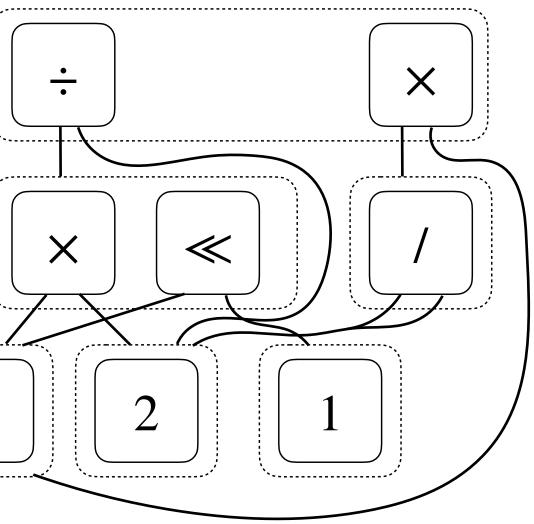


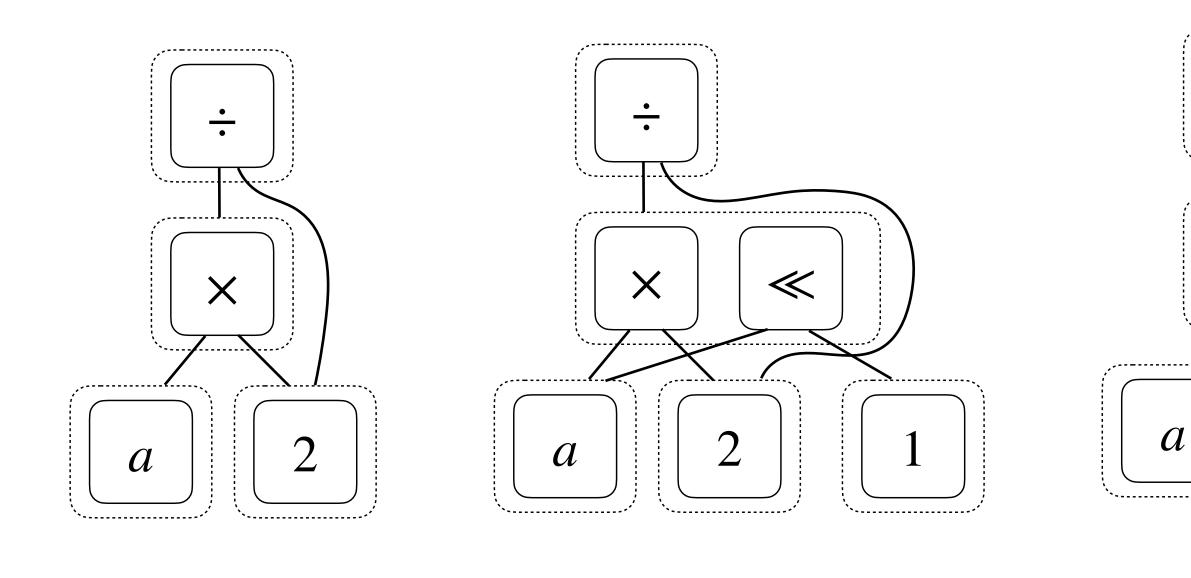


x \* 2 => x << 1

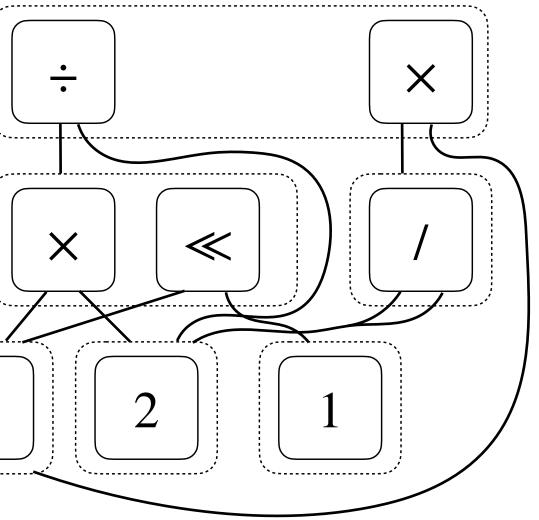


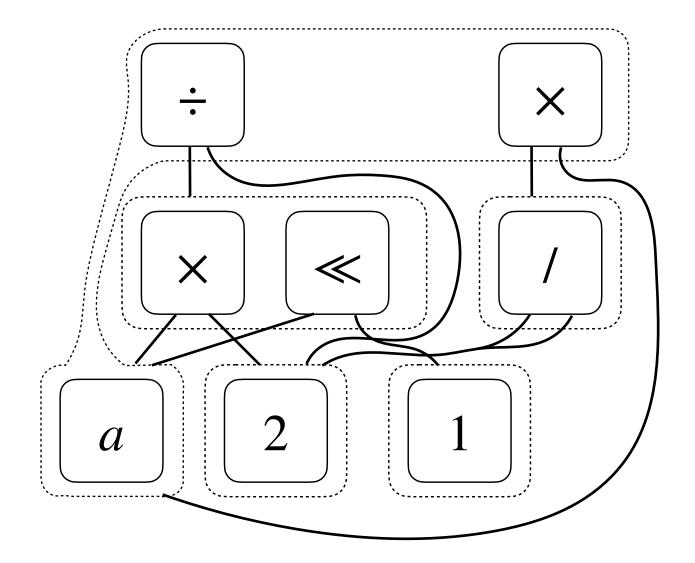
x \* 2 => x << 1 (x \* y) / z => x \* (y / z)



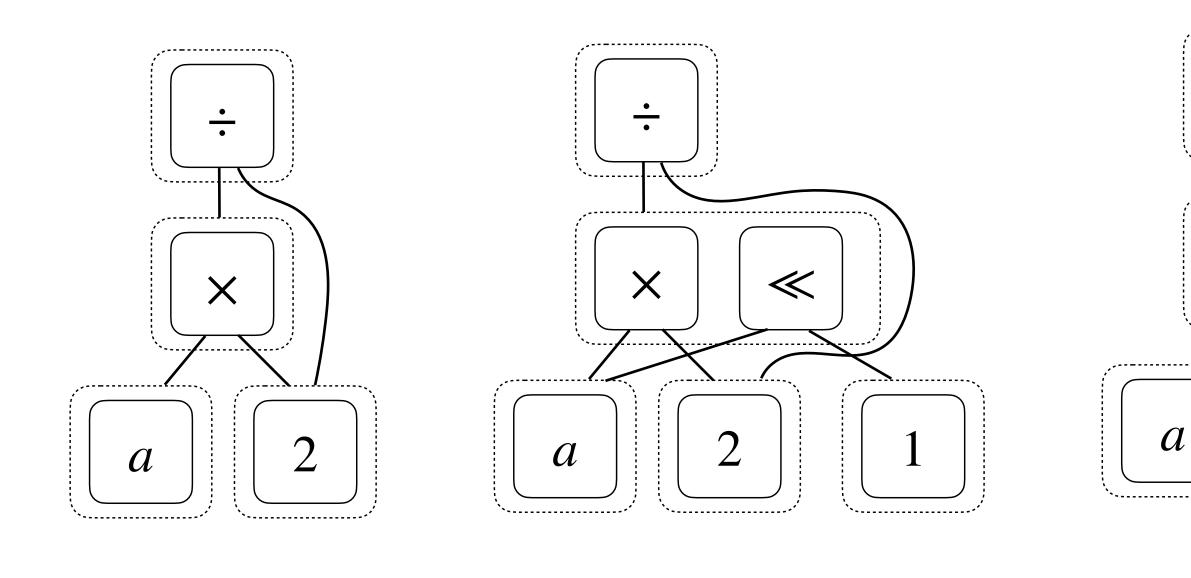


x \* 2 => x << 1 (x \* y) / z => x \* (y / z)

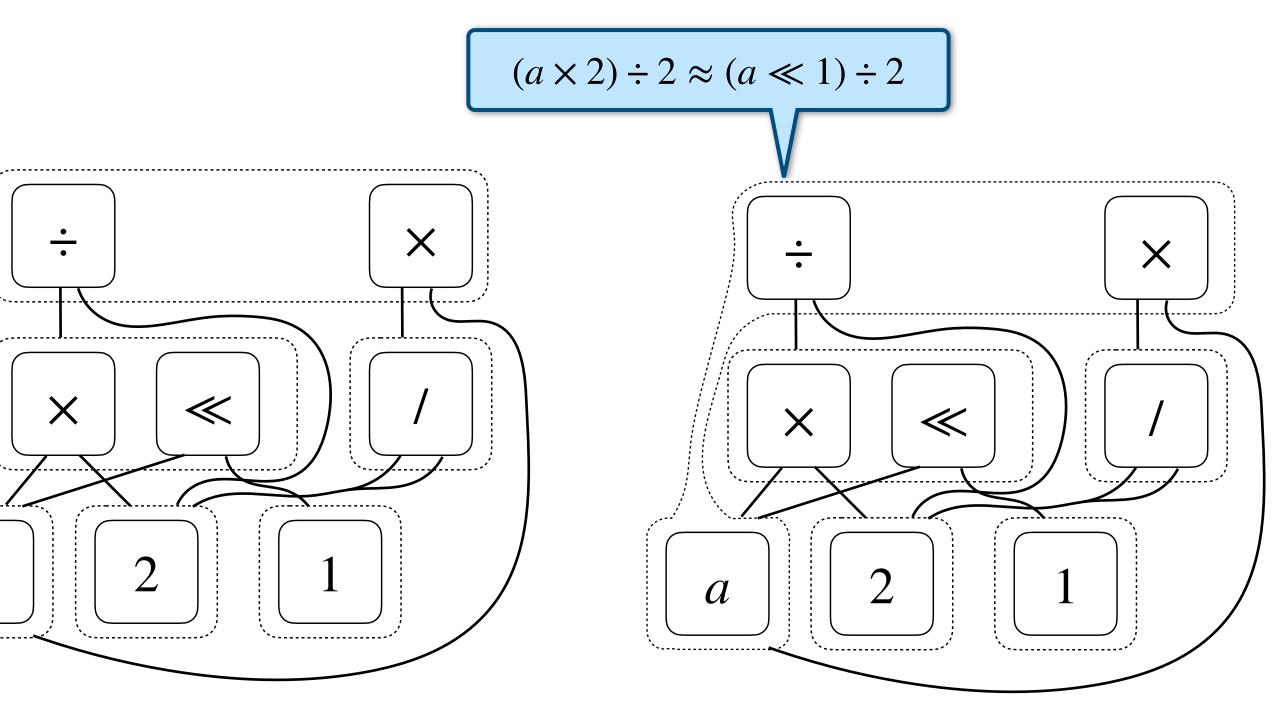




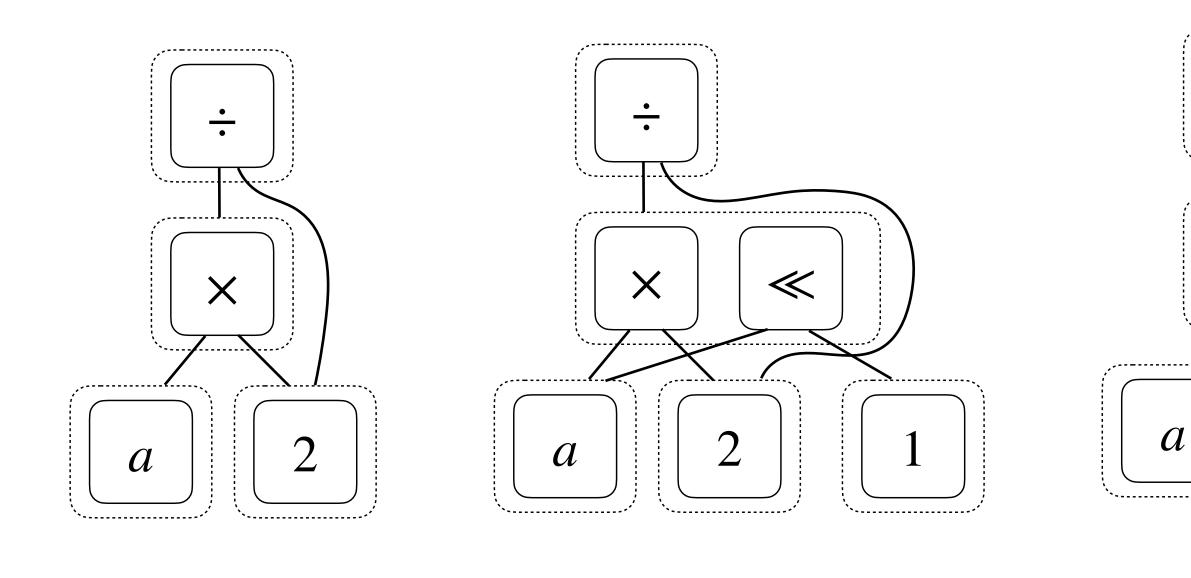
x / x => 1x \* 1 => x



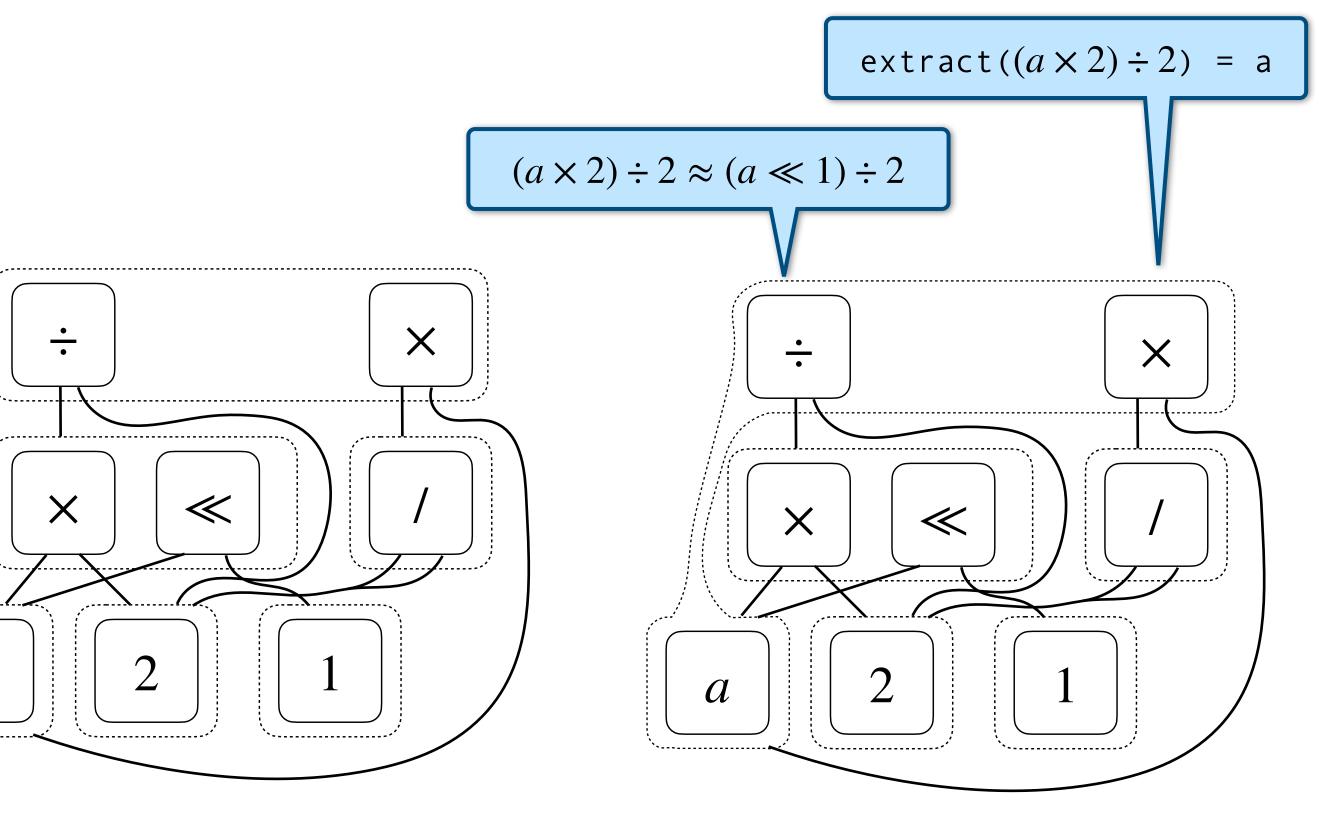
x \* 2 => x << 1 (x \* y) / z => x \* (y / z)



x / x => 1 x \* 1 => x



x \* 2 => x << 1 (x \*



(x \* y) / z = > x \* (y / z)

x / x => 1x \* 1 => x

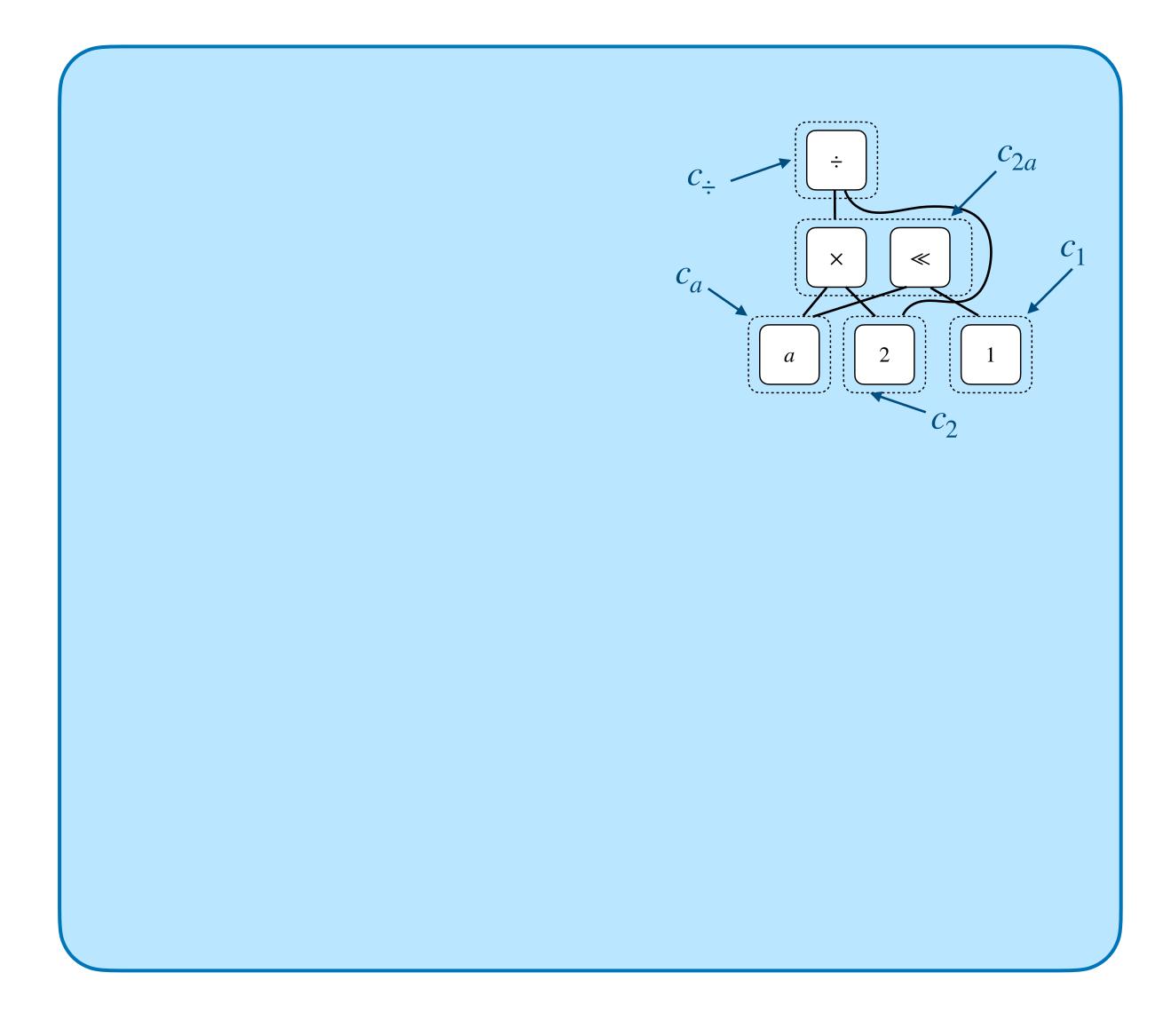
• Deterministic tree automaton  $\mathscr{A} = (Q, \Sigma, \Delta).$ 

We allow a tree automaton to be infinite.

Deterministic tree automaton

We allow a tree automaton to be infinite.

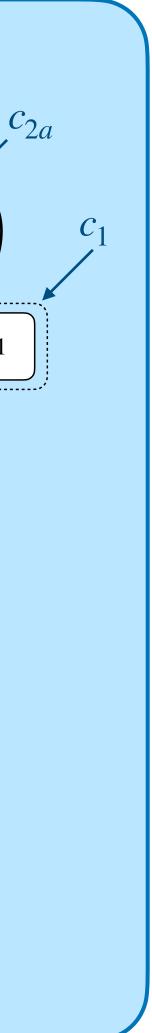
Deterministic tree automaton



We allow a tree automaton to be infinite.

Deterministic tree automaton

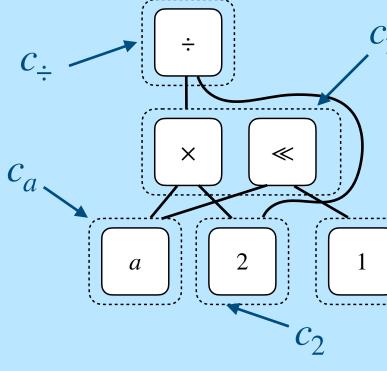
 $Q = \{c_1, c_2, c_a, c_{2a}, c_{\div}\},\$  $\ll$ a  $C_2$ 

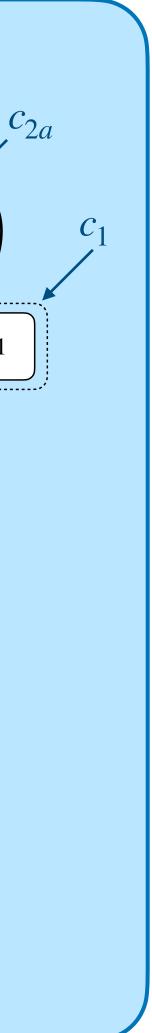


We allow a tree automaton to be infinite.

Deterministic tree automaton

$$\begin{split} Q &= \{c_1, c_2, c_a, c_{2a}, c_{\div}\}, \\ & \left\{ \begin{array}{l} a() \to c_a, \\ 1() \to c_1, 2() \to c_2 \\ \times (c_a, c_2) \to c_{2a}, \\ \ll (c_a, c_1) \to c_{2a} \\ \div (c_{2a}, c_2) \to c_{\div} \\ \end{split} \right. \end{split}$$

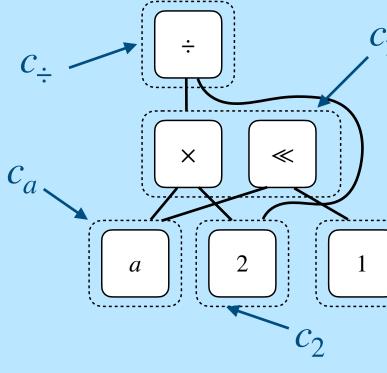


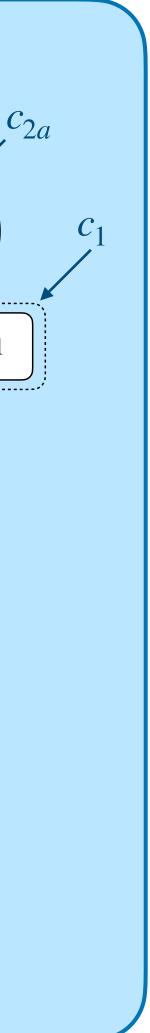


We allow a tree automaton to be infinite.

- Deterministic tree automaton  $\mathscr{A} = (Q, \Sigma, \Delta).$
- A term *t* is <u>represented</u> at  $q \in Q$  if  $t \to_{\mathscr{A}}^{*} q$ .

$$\begin{split} Q &= \{c_1, c_2, c_a, c_{2a}, c_{\div}\}, \\ & \left\{ \begin{array}{l} a() \to c_a, \\ 1() \to c_1, 2() \to c_2 \\ \times (c_a, c_2) \to c_{2a}, \\ \ll (c_a, c_1) \to c_{2a} \\ \div (c_{2a}, c_2) \to c_{\div} \\ \end{array} \right. \end{split}$$





We allow a tree automaton to be infinite.

- Deterministic tree automaton  $\mathscr{A} = (Q, \Sigma, \Delta).$
- A term *t* is <u>represented</u> at  $q \in Q$  if  $t \to_{\mathscr{A}}^{*} q$ .

$$Q = \{c_1, c_2, c_a, c_{2a}, c_{\div}\},$$

$$\Delta = \begin{cases} a() \rightarrow c_a, \\ 1() \rightarrow c_1, 2() \rightarrow c_2, \\ \times (c_a, c_2) \rightarrow c_{2a}, \\ \ll (c_a, c_1) \rightarrow c_{2a} \\ \div (c_{2a}, c_2) \rightarrow c_{\div} \end{cases}$$

$$(a \times 2) \div 2 \rightarrow^* (c_a \times c_2) \div c_2 \rightarrow c_{2a} \div c_2 \rightarrow c_{2a}$$



We allow a tree automaton to be infinite.

- Deterministic tree automaton  $\mathscr{A} = (Q, \Sigma, \Delta).$
- A term *t* is <u>represented</u> at  $q \in Q$  if  $t \to \mathcal{A}^* q$ .
- An <u>*E-graph</u> G* is a reachable deterministic</u> tree automaton.
  - E-classes = states, E-nodes = transitions

 $(a \times 2) \div 2 \rightarrow * (c_a \times c_2) \div c_2 \rightarrow c_{2a} \div c_2 \rightarrow c_{\dot{\pm}}$ 



We allow a tree automaton to be infinite.

- Deterministic tree automaton  $\mathscr{A} = (Q, \Sigma, \Delta).$
- A term *t* is <u>represented</u> at  $q \in Q$  if  $t \to_{\mathscr{A}}^{*} q$ .
- An  $\underline{E}$ -graph G is a reachable deterministic tree automaton.
  - E-classes = states, E-nodes = transitions
- [Kozen '93] The  $\underline{semantics}$  of an E-graph G is  $\approx_G$  :

 $t_1 \approx_G t_2 \Leftrightarrow \exists q \, . \, t_1 \to_G^* q \leftarrow_G^* t_2$ 

$$Q = \{c_1, c_2, c_a, c_{2a}, c_{\div}\},$$

$$\Delta = \begin{cases} a() \rightarrow c_a, \\ 1() \rightarrow c_1, 2() \rightarrow c_2, \\ \times (c_a, c_2) \rightarrow c_{2a}, \\ \ll (c_a, c_1) \rightarrow c_{2a} \\ \div (c_{2a}, c_2) \rightarrow c_{\div} \end{cases}$$

$$(a \times 2) \div 2 \rightarrow^* (c_a \times c_2) \div c_2 \rightarrow c_{2a} \div c_2 \rightarrow c_{2a} \div c_2 \rightarrow c_{2a}$$



L u

We allow a tree automaton to be infinite.

- Deterministic tree automaton  $\mathscr{A} = (Q, \Sigma, \Delta).$
- A term *t* is <u>represented</u> at  $q \in Q$  if  $t \to \mathcal{A}^* q$ .
- An <u>*E-graph</u> G* is a reachable deterministic</u> tree automaton.
  - E-classes = states, E-nodes = transitions
- [Kozen '93] The <u>semantics</u> of an E-graph G is  $\approx_G$  :

 $t_1 \approx_G t_2 \Leftrightarrow \exists q \, . \, t_1 \to_G^* q \leftarrow_G^* t_2$ 

 $(a \times 2) \div 2 \rightarrow * (c_a \times c_2) \div c_2 \rightarrow c_{2a} \div c_2 \rightarrow c_{\pm}$ 

 $\approx_G$  is defined as  $a \approx_G a, 2 \approx_G 2, \dots$  and two nontrivial identities

$$a \times 2 \approx_G a \ll 1$$
$$(a \times 2) \div 2 \approx_G (a \ll 1) \div 2$$



We allow a tree automaton to be infinite.

- Deterministic tree automaton  $\mathscr{A} = (O, \Sigma, \Delta).$
- A term *t* is <u>represented</u> at  $q \in Q$  if  $t \to \mathcal{A}^* q$ .
- An <u>*E-graph</u> G* is a reachable deterministic</u> tree automaton.
  - E-classes = states, E-nodes = transitions
- [Kozen '93] The <u>semantics</u> of an E-graph G is  $\approx_G$ :

$$t_1 \approx_G t_2 \Leftrightarrow \exists q \, . \, t_1 \to_G^* q \leftarrow_G^* t_2$$

•  $\approx_G$  is a partial congruence relation, or a PCR (= symmetric, transitive, and congruent).

 $(a \times 2) \div 2 \rightarrow * (c_a \times c_2) \div c_2 \rightarrow c_{2a} \div c_2 \rightarrow c_{\pm}$ 

 $\approx_G$  is defined as  $a \approx_G a, 2 \approx_G 2, \dots$  and two nontrivial identities

> $a \times 2 \approx_G a \ll 1$  $(a \times 2) \div 2 \approx_G (a \ll 1) \div 2$



• Tree automata <u>homomorphism</u>  $h: \mathscr{A} \to \mathscr{B}$ .



- Tree automata <u>homomorphism</u>  $h : \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .



- Tree automata <u>homomorphism</u>  $h : \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:



- Tree automata <u>homomorphism</u>  $h : \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:
  - $h_1, h_2: G_1 \to G_2$  implies  $h_1 = h_2$ .



- Tree automata <u>homomorphism</u>  $h : \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:
  - $h_1, h_2: G_1 \to G_2$  implies  $h_1 = h_2$ .
  - $\Box$  forms a partial order.



- Tree automata <u>homomorphism</u>  $h : \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:
  - $h_1, h_2: G_1 \to G_2$  implies  $h_1 = h_2$ .
  - $\Box$  forms a partial order.
  - $G_1 \sqsubseteq G_2$  implies  $\approx_{G_1} \subseteq \approx_{G_2}$ .



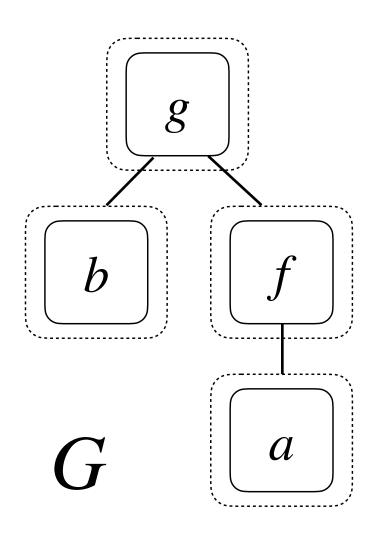
- Tree automata <u>homomorphism</u>  $h : \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:
  - $h_1, h_2: G_1 \to G_2$  implies  $h_1 = h_2$ .
  - $\Box$  forms a partial order.
  - $G_1 \sqsubseteq G_2$  implies  $\approx_{G_1} \subseteq \approx_{G_2}$ .
- <u>Congruence closure</u>: For arbitrary  $\mathscr{A}$ , there exists a unique minimal E-graph G such that  $\mathscr{A} \sqsubseteq G$ .



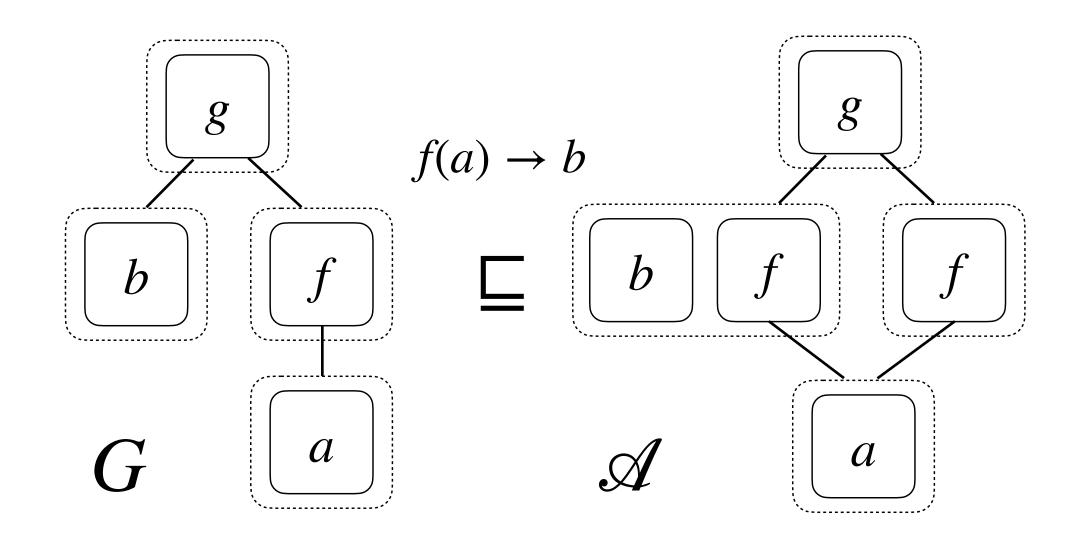
- Tree automata <u>homomorphism</u>  $h : \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:
  - $h_1, h_2: G_1 \to G_2$  implies  $h_1 = h_2$ .
  - $\Box$  forms a partial order.
  - $G_1 \sqsubseteq G_2$  implies  $\approx_{G_1} \subseteq \approx_{G_2}$ .
- <u>Congruence closure</u>: For arbitrary  $\mathscr{A}$ , there exists lacksquarea unique minimal E-graph G such that  $\mathscr{A} \sqsubseteq G$ .



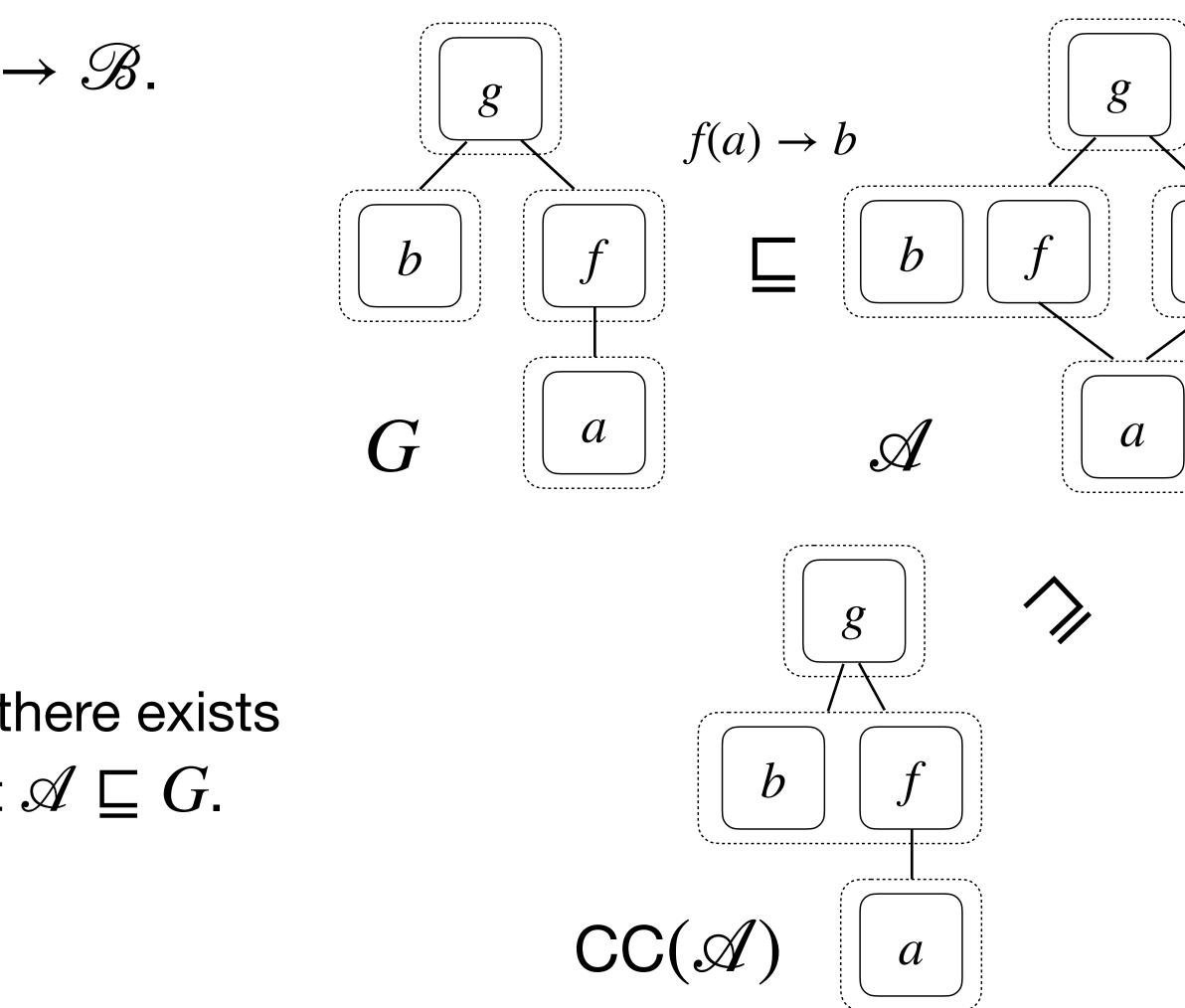
- Tree automata <u>homomorphism</u>  $h : \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:
  - $h_1, h_2: G_1 \to G_2$  implies  $h_1 = h_2$ .
  - $\Box$  forms a partial order.
  - $G_1 \sqsubseteq G_2$  implies  $\approx_{G_1} \subseteq \approx_{G_2}$ .
- <u>Congruence closure</u>: For arbitrary  $\mathscr{A}$ , there exists • a unique minimal E-graph G such that  $\mathscr{A} \sqsubseteq G$ .

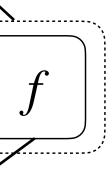


- Tree automata <u>homomorphism</u>  $h: \mathcal{A} \to \mathcal{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:
  - $h_1, h_2: G_1 \to G_2$  implies  $h_1 = h_2$ .
  - $\Box$  forms a partial order.
  - $G_1 \sqsubseteq G_2$  implies  $\approx_{G_1} \subseteq \approx_{G_2}$ .
- <u>Congruence closure</u>: For arbitrary  $\mathscr{A}$ , there exists lacksquarea unique minimal E-graph G such that  $\mathscr{A} \sqsubseteq G$ .



- Tree automata <u>homomorphism</u>  $h: \mathscr{A} \to \mathscr{B}$ .
- Define  $\mathscr{A} \sqsubseteq \mathscr{B}$  if  $\exists h : \mathscr{A} \to \mathscr{B}$ .
- When restricted to E-graphs:
  - $h_1, h_2: G_1 \to G_2 \text{ implies } h_1 = h_2.$
  - $\Box$  forms a partial order.
  - $G_1 \sqsubseteq G_2$  implies  $\approx_{G_1} \subseteq \approx_{G_2}$ .
- <u>Congruence closure</u>: For arbitrary  $\mathscr{A}$ , there exists a unique minimal E-graph G such that  $\mathscr{A} \sqsubseteq G$ .





#### Equality Saturation

# Equality Saturation

#### • The *immediate consequence operator*: (rule application $T_R$ defined in the paper)

# Equality Saturation

#### • The *immediate consequence operator*:

(rule application  $T_R$  defined in the paper)

$$\mathsf{ICO}_R = \mathsf{CC} \circ T_R$$

• The *immediate consequence operator*: (rule application  $T_R$  defined in the paper)

$$\mathsf{ICO}_R = \mathsf{CC} \circ T_R$$

• The *fixpoint semantics* of EqSat

• The *immediate consequence operator*: (rule application  $T_R$  defined in the paper)

$$\mathsf{ICO}_R = \mathsf{CC} \circ T_R$$

The <u>fixpoint semantics</u> of EqSat

$$\mathsf{EqSat}(R,G) = \bigsqcup_{i \ge 0} \mathsf{ICO}_R^{(i)}(G)$$

The *immediate consequence operator*: (rule application  $T_R$  defined in the paper)

$$\mathsf{ICO}_R = \mathsf{CC} \circ T_R$$

- The *fixpoint semantics* of EqSat  $\mathsf{EqSat}(R,G) = \bigsqcup_{i \ge 0} \mathsf{ICO}_R^{(i)}(G)$
- The *model semantics* of EqSat: EqSat(R, G) is a universal model of R and G(see the paper for defn. of universal models).

The *immediate consequence operator*: (rule application  $T_R$  defined in the paper)

$$\mathsf{ICO}_R = \mathsf{CC} \circ T_R$$

- The *fixpoint semantics* of EqSat  $\mathsf{EqSat}(R,G) = \bigsqcup_{i>0} \mathsf{ICO}_R^{(i)}(G)$
- The *model semantics* of EqSat: EqSat(R, G) is a universal model of R and G(see the paper for defn. of universal models).
- The two semantics coincide.

The *immediate consequence operator*: (rule application  $T_R$  defined in the paper)

$$\mathsf{ICO}_R = \mathsf{CC} \circ T_R$$

- The *fixpoint semantics* of EqSat  $\mathsf{EqSat}(R,G) = \bigsqcup_{i>0} \mathsf{ICO}_R^{(i)}(G)$
- The *model semantics* of EqSat: EqSat(R, G) is a universal model of R and G(see the paper for defn. of universal models).
- The two semantics coincide.

**Properties of EqSat** 

The *immediate consequence operator*: (rule application  $T_R$  defined in the paper)

$$\mathsf{ICO}_R = \mathsf{CC} \circ T_R$$

- The *fixpoint semantics* of EqSat  $\mathsf{EqSat}(R,G) = \bigsqcup_{i>0} \mathsf{ICO}_R^{(i)}(G)$
- The *model semantics* of EqSat: EqSat(R, G) is a universal model of R and G(see the paper for defn. of universal models).
- The two semantics coincide.

#### **Properties of EqSat**

#### • (Inflationary) $G \sqsubseteq EqSat(R, G)$ .

The *immediate consequence operator*: (rule application  $T_R$  defined in the paper)

$$\mathsf{ICO}_R = \mathsf{CC} \circ T_R$$

- The *fixpoint semantics* of EqSat  $\mathsf{EqSat}(R,G) = \bigsqcup_{i>0} \mathsf{ICO}_R^{(i)}(G)$
- The *model semantics* of EqSat: EqSat(R, G) is a universal model of R and G(see the paper for defn. of universal models).
- The two semantics coincide.

#### **Properties of EqSat**

- (Inflationary)  $G \sqsubseteq EqSat(R, G)$ .
- (Finite convergence) If EqSat(R, G) is finite, Equality Saturation converges in a finite number of steps.

### EqSat and Term Rewriting

## EqSat and Term Rewriting

- Let *R* be a TRS,  $s \in T_{\Sigma}$ , G = EqSat(R, s)
  - If  $s \to_R^* t$ , then  $s \approx_G t$ .
  - If  $s \approx_G t$ , then  $s \leftrightarrow_R^* t$ .

## **EqSat and Term Rewriting**

- Let *R* be a TRS,  $s \in T_{\Sigma}$ , G = EqSat(R, s)
  - If  $s \to_R^* t$ , then  $s \approx_G t$ .
  - If  $s \approx_G t$ , then  $s \leftrightarrow_R^* t$ .
- If *R* is bidirectional,  $\rightarrow_R^*$ ,  $\approx_G^*$ ,  $\leftrightarrow_R^*$  coincide.
  - In this case, EqSat semi-decides the word problem.

- The Chase
  - A generalization of Datalog.
  - Used in data integration and query optimization.
  - Two variants: Skolem Chase, Standard Chase.

timization. d Chase.

- The Chase
  - A generalization of Datalog.
  - Used in data integration and query optimization.
  - Two variants: Skolem Chase, Standard Chase.
- Skolem Chase  $\leq$  EqSat.

timization. d Chase.

- The Chase  $\bullet$ 
  - A generalization of Datalog.
  - Used in data integration and query optimization.
  - Two variants: Skolem Chase, Standard Chase.
- Skolem Chase  $\leq$  EqSat.
- EqSat  $\leq$  Standard Chase.
  - EqSat terminates ⇔
  - $\exists$  a finite run of the encoding chase  $\Leftrightarrow$ lacksquare
  - All <u>EGD-fair</u> runs of the encoding chase are finite.  $\bullet$

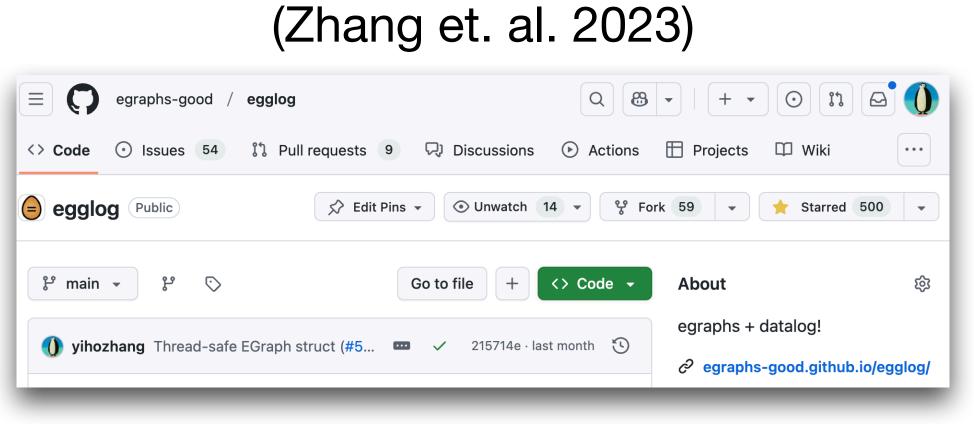
- The Chase
  - A generalization of Datalog.
  - Used in data integration and query optimization.  $\bullet$
  - Two variants: Skolem Chase, Standard Chase.
- Skolem Chase  $\leq$  EqSat.
- EqSat  $\leq$  Standard Chase.  $\bullet$ 
  - EqSat terminates ⇔
  - $\exists$  a finite run of the encoding chase  $\Leftrightarrow$
  - All <u>EGD-fair</u> runs of the encoding chase are finite.  $\bullet$

The chase sequence needs to apply EGD often enough

- The Chase lacksquare
  - A generalization of Datalog.
  - Used in data integration and query optimization.
  - Two variants: Skolem Chase, Standard Chase.
- Skolem Chase  $\leq$  EqSat.  $\bullet$
- EqSat  $\leq$  Standard Chase.  $\bullet$ 
  - EqSat terminates  $\Leftrightarrow$ lacksquare
  - $\exists$  a finite run of the encoding chase  $\Leftrightarrow$  $\bullet$
  - All <u>EGD-fair</u> runs of the encoding chase are finite.  $\bullet$

The chase sequence needs to apply EGD often enough

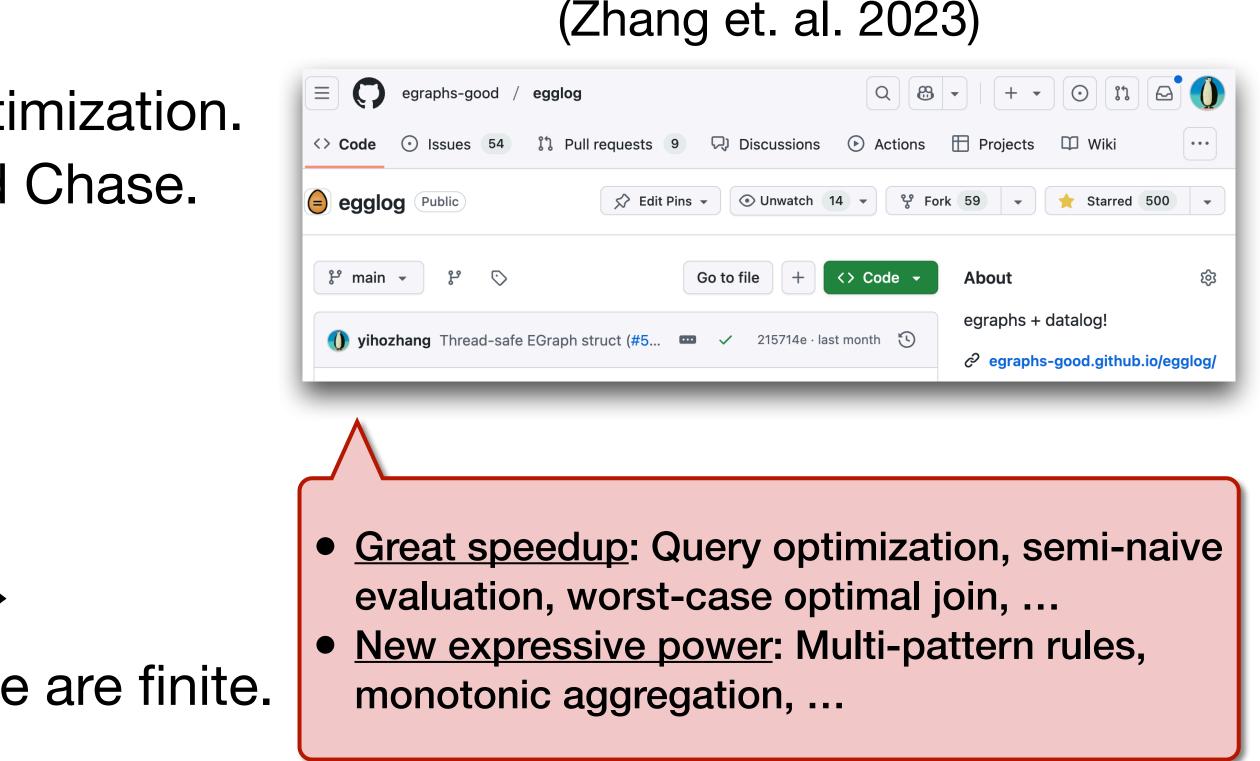
#### We build an entire system out of this idea! (Zhang et. al. 2023)



- The Chase
  - A generalization of Datalog.
  - Used in data integration and query optimization.
  - Two variants: Skolem Chase, Standard Chase.
- Skolem Chase  $\leq$  EqSat.
- EqSat  $\leq$  Standard Chase.
  - EqSat terminates ⇔
  - $\exists$  a finite run of the encoding chase  $\Leftrightarrow$
  - All EGD-fair runs of the encoding chase are finite.

The chase sequence needs to apply EGD often enough

#### We build an entire system out of this idea! (Zhang et. al. 2023)



## **Termination Theorem**

- (Single-instance) Does EqSat terminate with for a single term t? • Recursive enumerable (R.E.) – complete.
- (All-E-graph) Does EqSat terminate for all E-graph G?
  - R.E.-hard.
- (All-term) Does EqSat terminate for all  $t \in T_{\Sigma}$ ?
  - $\Pi_2$ -complete.

## **Termination Theorem**

- (Single-instance) Does EqSat terminate with for a single term t? • Recursive enumerable (R.E.) – complete.
- (All-E-graph) Does EqSat terminate for all E-graph G?
  - R.E.-hard.
- (All-term) Does EqSat terminate for all  $t \in T_{\Sigma}$ ?
  - $\Pi_2$ -complete.

Strictly harder than R.E.

## **Termination Theorem**

- (Single-instance) Does EqSat terminate with for a single term t? • Recursive enumerable (R.E.) – complete.
- (All-E-graph) Does EqSat terminate for all E-graph G?
  - R.E.-hard.
- (All-term) Does EqSat terminate for all  $t \in T_{\Sigma}$ ?
  - $\Pi_2$ -complete.

Strictly harder than R.E.

More details in the paper!

### Summary

- The fixpoint and model semantics of Equality Saturation
- Connections to Term Rewriting and the Chase
- Undecidability of Termination
- Open problems
  - Extraction
  - Provenance

#### <u>Also check out the egglog system!</u> <u>GitHub: @egraphs-good/egglog</u>

$\equiv$ $\bigcirc$ egraphs-good / eggle	og	Q	- + - 0	ii 🗗 🚺
<> Code ① Issues 54 រ៉ា រ	Pull requests 9 🖓 Discussions	Actions	🗄 Projects 🛛 🕮 V	Viki
egglog Public	S Edit Pins - O Unwatch	14 - 양 Fork	59 🔹 🌟 S	tarred 500 -
ᢞ main ▾ ᢞ ♡	Go to file +	<> Code -	About	ŝ
<b>()</b> yihozhang Thread-safe EGrap	h struct ( <b>#5  ✓</b> 215714e · la	ast month 🕓	egraphs + datalog	-
		1		

