DATA DRIVEN APPROXIMATION OF ABSTRACT TRANSFORMERS

Shaurya Gomber
1st yr MS CS
UIUC

Contact: sgomber2@illinois.edu

Topics

1. Abstract Transformers (Quick Recap)
2. Motivation
3. Problem Setting
4. Technique
5. Future Work

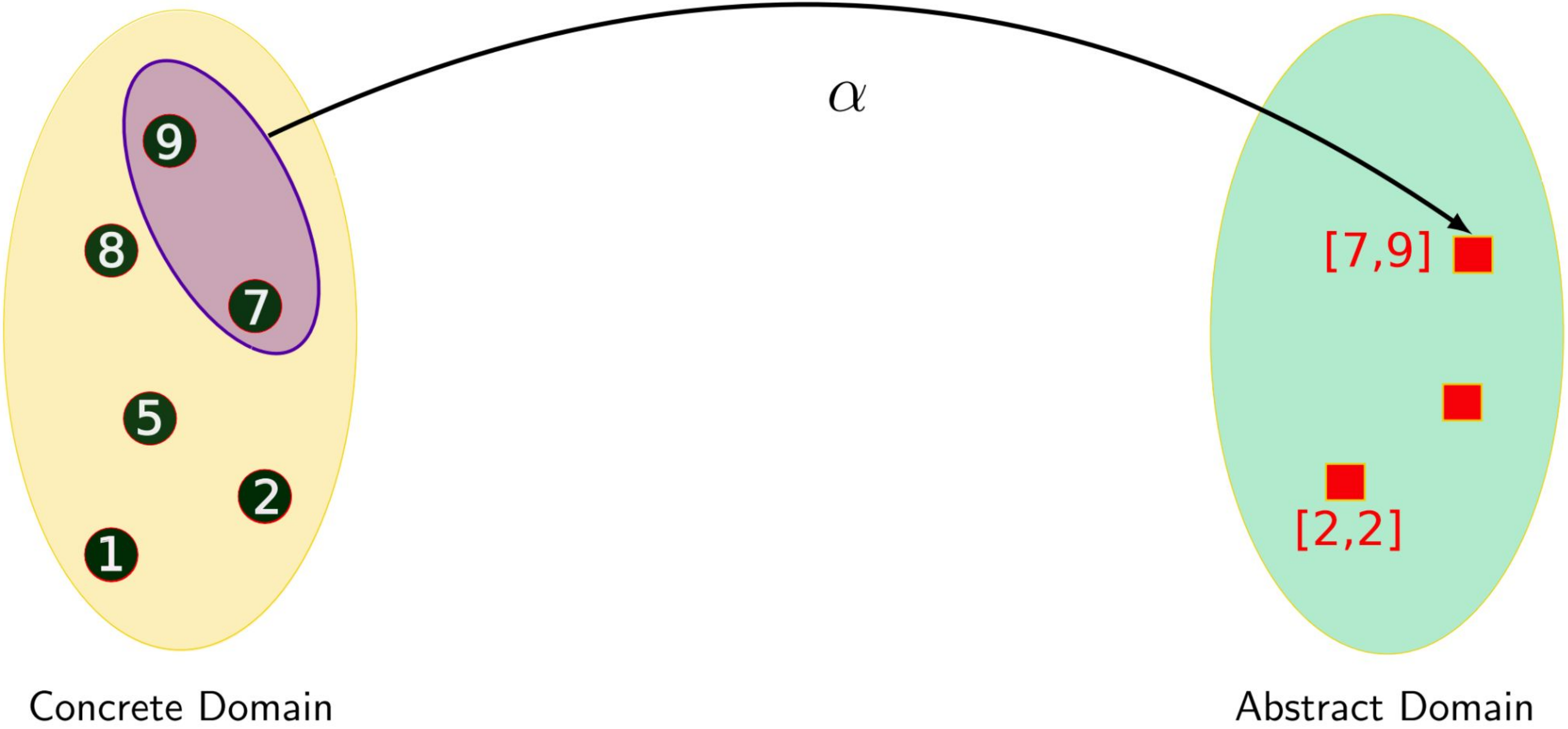
ABSTRACT TRANSFORMERS

Abstract Domains

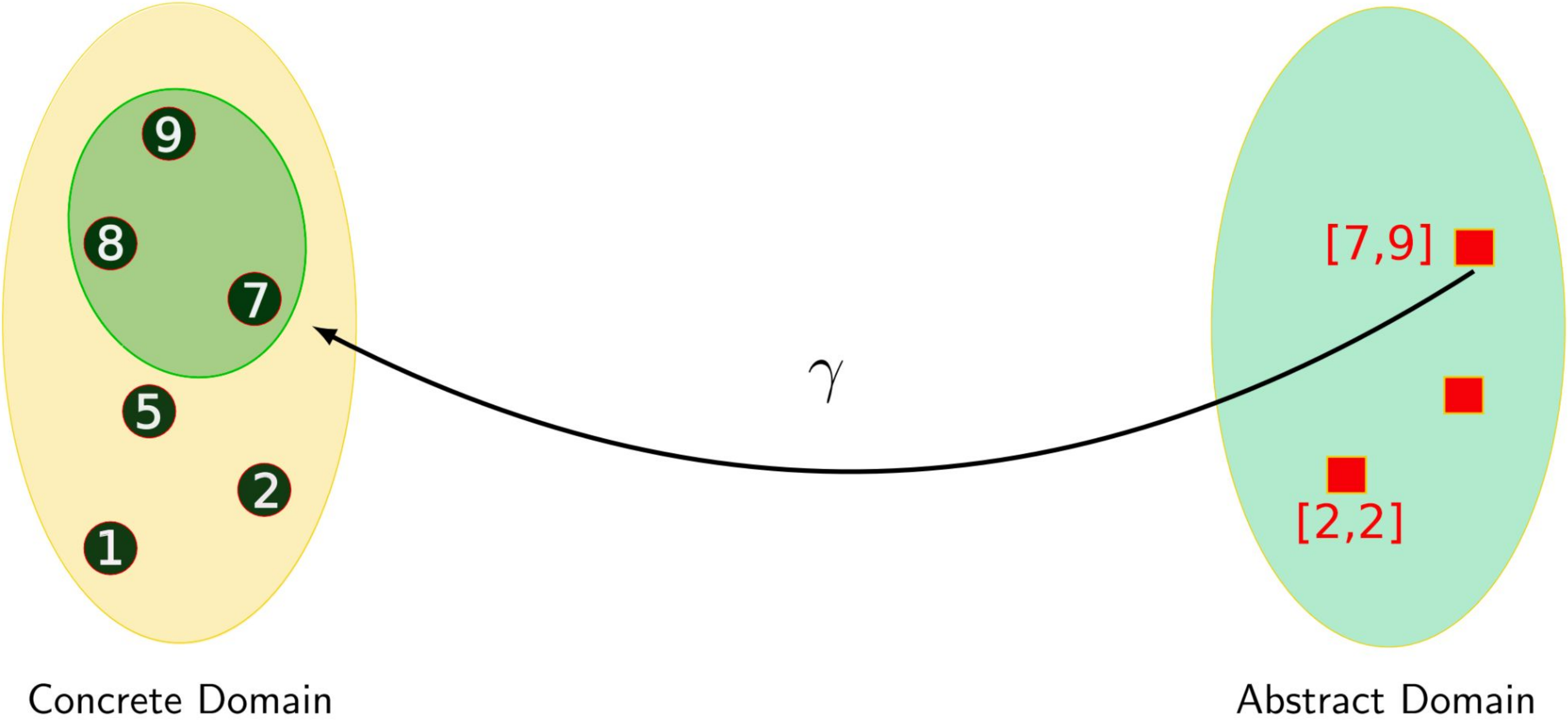
- Domain of values different from our program states.
- Used to keep track of the program states *succinctly*.
- Some examples: Interval, Zonotopes, Octagon, Polyhedra

Concrete Domain Abstract Domain

Abstraction Function

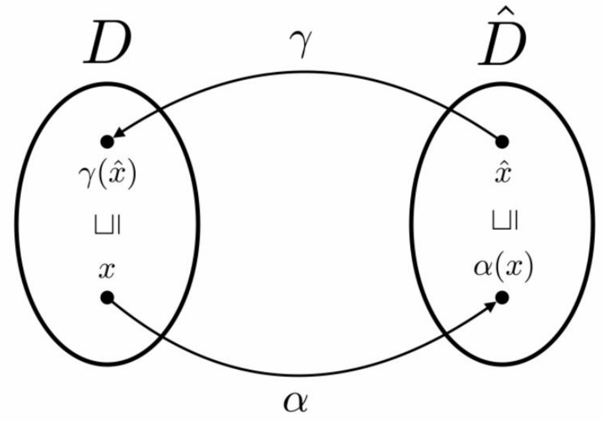


Concretization Function



Galois Connection

$$\forall x \in D, \forall \hat{x} \in \hat{D}.\ \alpha(x) \sqsubseteq \hat{x} \Leftrightarrow x \sqsubseteq \gamma(\hat{x})$$



Intuitively, this says that α, γ respect the orderings of $D, \hat{D}$

Introduction

- Consider: + operation, code line z = x + y, Interval Domain

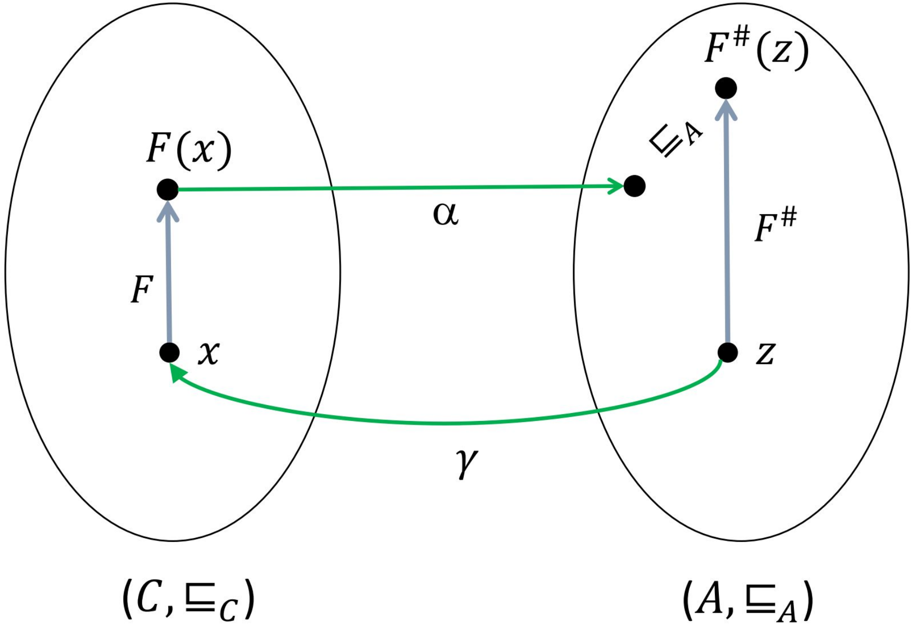
- If $x^\#$ = [a, b] and $y^\#$ = [c, d], we need a operator $+^\#$ that gives us $z^\#$

  $z^\# = x^\# +^\# y^\# = [a, b] +^\# [c, d] = [a+b, c+d]$
- We call $+^\#$ the *abstract transformer* for +

- Abstract Transformers are needed for:
  - All operations possible in the language (+, -, abs)
  - Lattice operations like join and meet (to handle if-else, while etc.)

Soundness

$$\forall z \in A.\, \alpha\Big(F\big(\gamma(z)\big)\Big) \sqsubseteq_A F^{\#}(z)$$



- **Necessary** condition for transformer correctness.
- If $[a, b] +^{\#} [c,d] = [e, f]$, then $+^{\#}$ is sound if:

  $\forall\, x \in [a, b], \forall\, y \in [c, d] \Rightarrow x + y \in [e, f]$

Precision

- Important for the practical applicability of abstract interpretation.
- Can be thought of as the measure of succinctness of the transformer's output.

Consider $[a, b] +^{\#} [c,d] = [e, f]$, $+^{\#}$ is sound if $\forall\ x \in [a, b],\ \forall\ y \in [c, d] \Rightarrow x + y \in [e, f]$

Now consider the following possible transformers:

	SOUND?	PRECISE?
$[-\infty, \infty]$		
$[a + c - 5, b + d + 6]$		
$[a + c + 1, b + d]$		
$[a + c, b + d]$		

Precision

- Important for the practical applicability of abstract interpretation.
- Can be thought of as the measure of succinctness of the transformer's output.

Consider $[a, b] +^{\#} [c,d] = [e, f]$, $+^{\#}$ is sound if $\forall\ x \in [a, b], \forall\ y \in [c, d] \Rightarrow x + y \in [e, f]$

Now consider the following possible transformers:

	SOUND?	PRECISE?
$[-\infty, \infty]$	**YES**	
$[a + c - 5, b + d + 6]$		
$[a + c + 1, b + d]$		
$[a + c, b + d]$		

Precision

- Important for the practical applicability of abstract interpretation.
- Can be thought of as the measure of succinctness of the transformer's output.

Consider $[a, b] +^{\#} [c,d] = [e, f]$, $+^{\#}$ is sound if $\forall\ x \in [a, b], \forall\ y \in [c, d] \Rightarrow x + y \in [e, f]$

Now consider the following possible transformers:

	SOUND?	PRECISE?
$[-\infty, \infty]$	**YES**	**NO**
$[a + c - 5, b + d + 6]$		
$[a + c + 1, b + d]$		
$[a + c, b + d]$		

Precision

- Important for the practical applicability of abstract interpretation.
- Can be thought of as the measure of succinctness of the transformer's output.

Consider $[a, b] +^{\#} [c,d] = [e, f]$, $+^{\#}$ is sound if $\forall\ x \in [a, b], \forall\ y \in [c, d] \Rightarrow x + y \in [e, f]$

Now consider the following possible transformers:

	SOUND?	PRECISE?
$[-\infty, \infty]$	YES	NO
$[a + c - 5, b + d + 6]$	YES	
$[a + c + 1, b + d]$		
$[a + c, b + d]$		

Precision

- Important for the practical applicability of abstract interpretation.
- Can be thought of as the measure of succinctness of the transformer's output.

Consider $[a, b] +^{\#} [c,d] = [e, f]$, $+^{\#}$ is sound if $\forall\ x \in [a, b], \forall\ y \in [c, d] \Rightarrow x + y \in [e, f]$

Now consider the following possible transformers:

	SOUND?	PRECISE?
$[-\infty, \infty]$	YES	NO
$[a + c - 5, b + d + 6]$	YES	NO
$[a + c + 1, b + d]$		
$[a + c, b + d]$		

Precision

- Important for the practical applicability of abstract interpretation.
- Can be thought of as the measure of succinctness of the transformer's output.

Consider $[a, b] +^{\#} [c,d] = [e, f]$, $+^{\#}$ is sound if $\forall\ x \in [a, b], \forall\ y \in [c, d] \Rightarrow x + y \in [e, f]$

Now consider the following possible transformers:

	SOUND?	PRECISE?
$[-\infty, \infty]$	YES	NO
$[a + c - 5, b + d + 6]$	YES	NO
$[a + c + 1, b + d]$	NO	<don't care>
$[a + c, b + d]$		

Precision

- Important for the practical applicability of abstract interpretation.
- Can be thought of as the measure of succinctness of the transformer's output.

Consider $[a, b] +^{\#} [c,d] = [e, f]$, $+^{\#}$ is sound if $\forall\ x \in [a, b], \forall\ y \in [c, d] \Rightarrow x + y \in [e, f]$

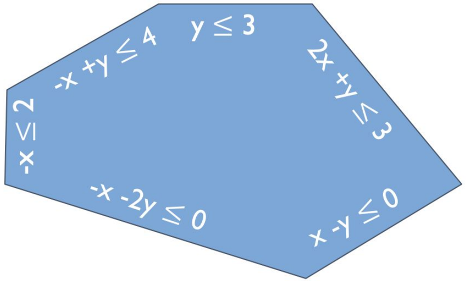
Now consider the following possible transformers:

	SOUND?	PRECISE?
$[-\infty, \infty]$	YES	NO
$[a + c - 5, b + d + 6]$	YES	NO
$[a + c + 1, b + d]$	NO	<don't care>
$[a + c, b + d]$	YES	YES

WHY APPROXIMATE THE TRANSFORMERS?

Polyhedra Domain

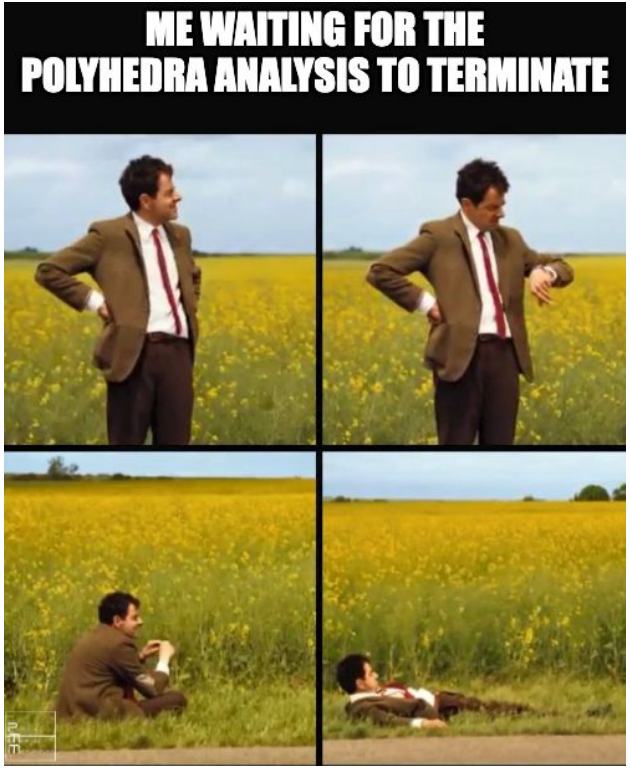
Represents linear constraints between program variables



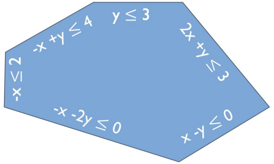
- Very powerful as it maintains all the complex relations between program variables.
- Stronger than the non-relational and weakly relational domains that we have seen in this course.
- Able to prove complex properties like $y < 2x$ easily.

Q. If this is so powerful, why do we even need other domains?

Polyhedra Domain Issue



Ans. Because the sound and most precise transformers for many of its operations are **computationally very expensive!**



Consider this:

- After an if-else block, there will be 2 such polyhedras.
- That is, we need to join two such figures.
- Time complexity: exponential in #vertices and #edges.
- Have heard of a case where it took ~3 days to compute one such join.

So, what next?

HOW TO APPROXIMATE SUCH COSTLY TRANSFORMERS?

Example

- Interval Domain
- **Goal:** Approximate the transformer for the abs method.

$$\mathtt{abs}^{\sharp}(\mathtt{a}) = [\mathtt{max}(\mathtt{max}(0, \mathtt{a.l}), -\mathtt{a.r}),\ \mathtt{max}(-\mathtt{a.l}, \mathtt{a.r})].$$

a.l : Lower bound of the specified interval a
a.r : Upper bound of the specified interval a

Setting: What do we have

- **Data:** Input Output examples (yes, we will have to run the costly transformer once to get the dataset)

(abs$^{\#}$([l, r]) = [absl, absr])

l	r	absl	absr
-2417.2257	8425.0984	0	8425.0984
9395.7928	9454.3504	9395.7928	9454.3504
-5975.7502	-2391.1638	2391.1638	5975.7502

- **Soundness Constraint:** forall x, (l <= x <= r) => (absl <= abs(x) <= absr)
- **Precision Measure:** | absr - absl |

PROBLEM SETTING

Goal

Use:

1. Data
2. Soundness constraint
3. Precision measure

to approximate the abstract transformer!

Q1. Why approximate the transformer?
Ans. An efficient (quick) way to get the transformed values (which are sound *most of the times*)

Q2. We have data and want to learn something out of it, whom do we call?

Goal

Use:

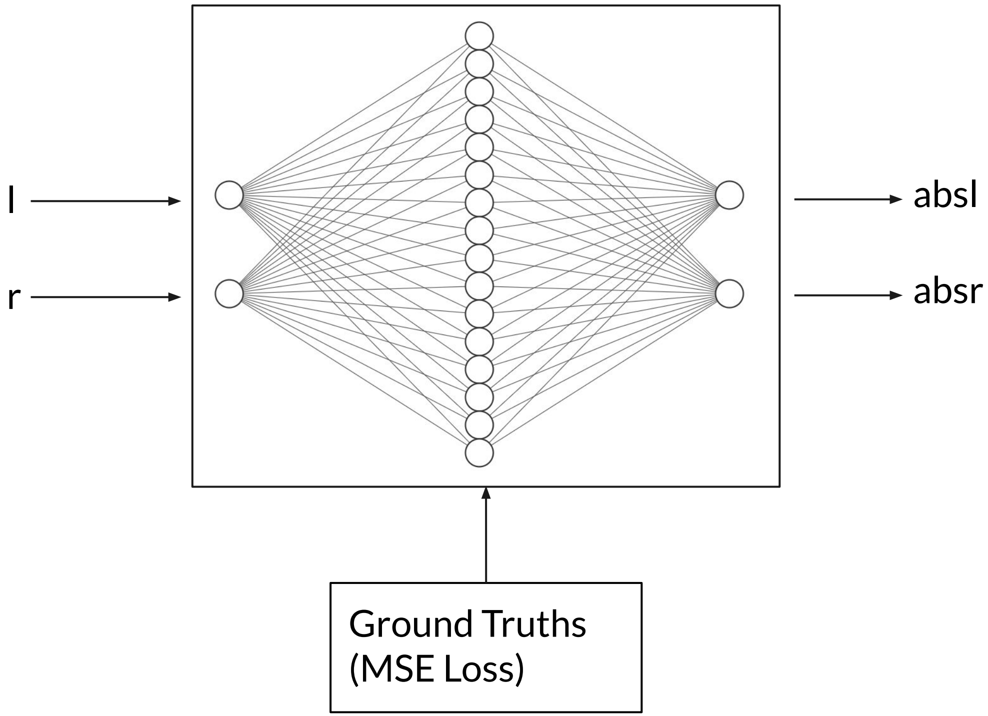
1. Data
2. Soundness constraint
3. Precision measure

to approximate the abstract transformer!

Q1. Why approximate the transformer?
Ans. An efficient (quick) way to get the transformed values (which are sound *most of the times*)

Q2. We have data and want to learn something out of it, whom do we call?
Ans. Neural Networks (obviously!)

Naive Way



- Give it only the entire dataset
- Ask it to memorize
- Penalize it with MSE loss

Problem with Naive Way

- Network asked only to memorize the data.
- Is oblivious to the soundness requirement.
- Learns a function in its hypothesis space that reduces the MSE loss well (so precision is fine).
- But it can not be used as a transformer because of the poor soundness results!

l	r	absl	absr	Sound?
7101.075	9944.418	7101.167	9943.465	NO (7101.075)
4796.38	8357.237	4796.295	8356.702	NO (8357.237)
-2620.969	2744.13	-1.009	2744.396	YES
-434.58	721.211	-0.380	722.148	YES
-3504.81	-320.015	319.993	3506.315	YES
8486.121	8783.284	8486.524	8782.35	NO (8783.284)
2175.55	9850.599	2174.945	9850.119	NO (9850.599)
9762.237	9903.25	9760.715	9902.053	NO (9903.25)
4894.421	9665.724	4894.224	9665.011	NO (9665.724)
8864.991	9757.781	8865.355	9756.688	NO (7101.075)

Results on 10 random inputs (Soundness measure: 30%)

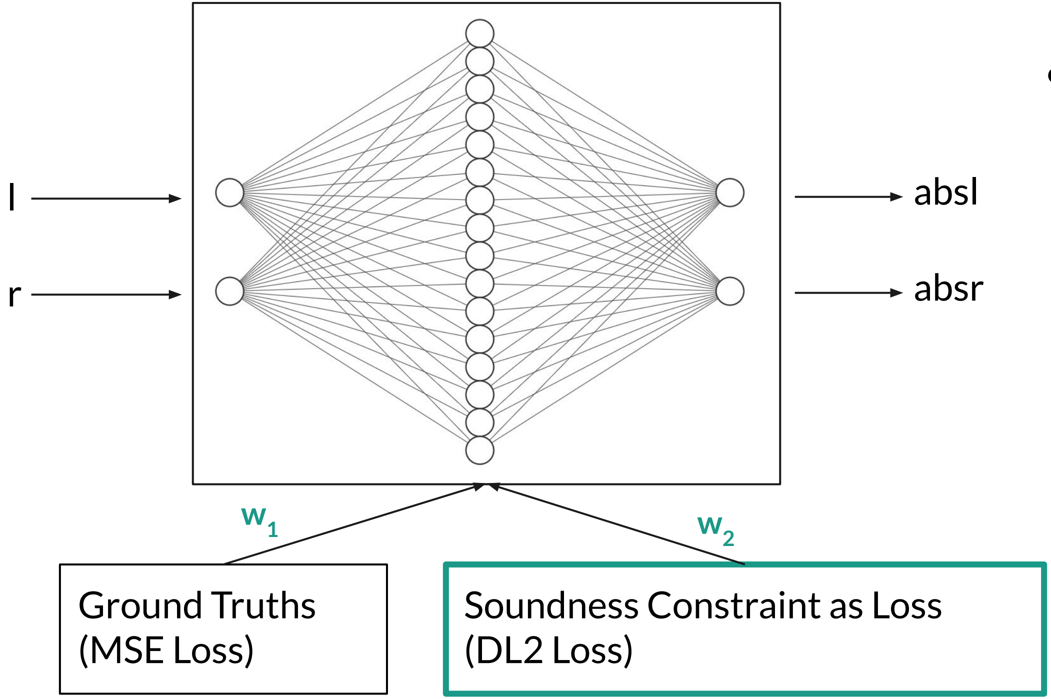
Problem with Naive Way

Training Logs:

```
=============== Testing ===============n
Evaluation on test set:
[0] MSE Loss: 20.487, Precision Loss: 30.566, Constr Loss: 0.179, Constr Accuracy: 0.6562
[10] MSE Loss: 0.721, Precision Loss: 1.947, Constr Loss: 0.052, Constr Accuracy: 0.8750
Avg. MSE Loss: 9.023, Avg. Precision Loss: 25.254, Avg. Constr Loss: 0.148, Avg. Constr Accuracy: 0.7548, Total Time: 2.1267
---------------------------------------------------

=============== Measuring Soundess ===============n
Soundness %: 37.5
625 Counterexamples found for 1000 runs:
Counter-example 1: {'model_input': [4791.904, 5210.99], 'model_output': [4792.1220703125, 5210.85595703125], 'ceg': 598988/125}
Counter-example 2: {'model_input': [1926.8563, 9833.2735], 'model_output': [1926.2178955078125, 9832.8193359375], 'ceg': 19666547/2000}
Counter-example 3: {'model_input': [-9243.9309, 3429.8234], 'model_output': [-1.4633784294128418, 9241.705078125], 'ceg': -4732265/512}
Counter-example 4: {'model_input': [5849.5943, 9480.1439], 'model_output': [5849.5517578125, 9479.3671875], 'ceg': 94801439/10000}
Counter-example 5: {'model_input': [-9422.8131, 4609.3722], 'model_output': [-1.5795893669128418, 9421.3037109375], 'ceg': -9648439/1024}
```

Our tool



We use the differentiable loss presented in the DL2: Deep Learning with Differential Logic paper (Fischer et. al).

- Converted Soundness Constraint to DL2 loss using these rules:

$$\mathcal{L}(t \leq t') := \max(t - t', 0),$$
$$\mathcal{L}(t \neq t') := \xi \cdot [t = t'].$$

$$\mathcal{L}(\varphi' \wedge \varphi'') := \mathcal{L}(\varphi') + \mathcal{L}(\varphi''),$$
$$\mathcal{L}(\varphi' \vee \varphi'') := \mathcal{L}(\varphi') \cdot \mathcal{L}(\varphi'').$$

- Adam optimizer & Projected Gradient Descent (PGD) used to solve the optimization problem.
- $w_1 = 1$; $w_2 = 2000$-3000 as we need to enforce soundness (almost as a hard constraint)

l	r	absl	absr	Sound?
7101.075	9944.418	7101.167	9943.465	NO (7101.075)
4796.38	8357.237	4796.295	8356.702	NO (8357.237)
-2620.969	2744.13	-1.009	2744.396	YES
-434.58	721.211	-0.380	722.148	YES
-3504.81	-320.015	319.993	3506.315	YES
8486.121	8783.284	8486.524	8782.35	NO (8783.284)
2175.55	9850.599	2174.945	9850.119	NO (9850.599)
9762.237	9903.25	9760.715	9902.053	NO (9903.25)
4894.421	9665.724	4894.224	9665.011	NO (9665.724)
8864.991	9757.781	8865.355	9756.688	NO (7101.075)

Results on 10 random inputs (Soundness measure: 30%)

Results from our tool

- Network asked to:
  - Memorize data (MSE Loss)
  - Follow Soundness Constraint (DL2 loss)

- Learnt network can be used (with some modifications) as a transformer because of the good soundness results!

- Precision can be improved? Yes, maybe!

l	r	absl	absr	Sound?
1239.651	4900.376	1236.163	4931.959	YES
-2335.049	4984.283	-38.855	5018.712	YES
-24.584	4452.161	30.151	4468.344	NO (0)
-9360.782	4349.862	-129.254	9522.770	YES
-7158.819	1744.48	-99.675	7291.221	YES
-1247.04	2484.397	-21.911	2503.420	YES
-6495.725	3592.935	-90.908	6606.015	YES
-4720.175	5626.187	-66.002	5681.784	YES
-1631.28	2022.999	-25.116	2043.962	YES
-751.029	-210.949	185.281	770.471	YES

Results on 10 random inputs (Soundness measure: 90%)

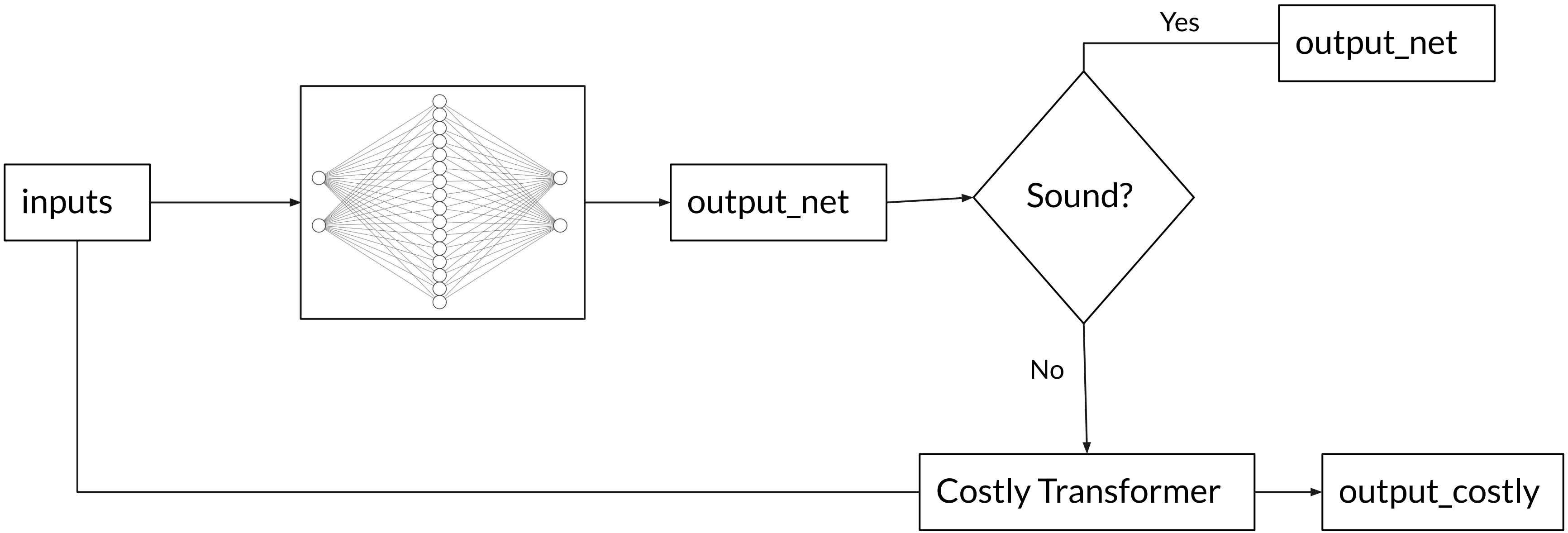
Results from our tool

Training Logs:

```
=============== Testing ===============n
Evaluation on test set:
[0] MSE Loss: 14172.818, Precision Loss: 47378.035, Constr Loss: 0.000, Constr Accuracy: 1.0000
[10] MSE Loss: 7047.575, Precision Loss: 23012.430, Constr Loss: 0.000, Constr Accuracy: 1.0000
Avg. MSE Loss: 12844.427, Avg. Precision Loss: 44800.696, Avg. Constr Loss: 0.014, Avg. Constr Accuracy: 0.9976, Total Time: 2.0774
--------------------------------------------------

=============== Measuring Soundess ===============n
Soundness %: 99.0
10 Counterexamples found for 1000 runs:
Counter-example 1: {'model_input': [28.6347, 9272.577], 'model_output': [91.18456268310547, 9304.20703125], 'ceg': 286347/10000}
Counter-example 2: {'model_input': [-149.6685, 7735.8132], 'model_output': [18.27129554748535, 7761.3955078125], 'ceg': 345425910949707/2000000000000}
Counter-example 3: {'model_input': [-36.2909, 2716.2896], 'model_output': [10.644037246704102, 2726.962158203125], 'ceg': 4822018623352051/500000000000000}
Counter-example 4: {'model_input': [-74.1385, 9136.8622], 'model_output': [55.91088104248047, 9167.3525390625], 'ceg': 5491088104248047/100000000000000}
Counter-example 5: {'model_input': [-165.2073, 8350.1459], 'model_output': [18.6539249420166, 8377.5400390625], 'ceg': 88269624710083/5000000000000}
```

Endgame



Applicability of this tool would depend upon the percentage of sound answers suggested by the network!!

FUTURE WORK

Future Work

- Add mechanisms to enforce Precision while training.

  This means that we would have three kinds of losses:
  - MSE Loss from data
  - Loss from soundness constraint
  - Loss from precision measure

- Use the framework on serious examples like Polyhedra join.

  This would involve:
  - Collecting training data
  - Implementing polyhedra join soundness as a DL2 constraint
  - Adding some notion of precision
  - Incorporating it with present verifiers to check efficacy on real world benchmarks.

Thanks!

Backup Slides

404 Not Found