Automated selection of grounding algorithm in Answer Set Programming

Marco Maratea[1], Luca Pulina[2] and Francesco Ricca[3]

[1] DIBRIS, University of Genova
[2] POLCOMING, University of Sassari
[3] Dep. of Matematichs and Informatics, University of Calabria

13th Conference of the Italian Association for Artificial Intelligence (AI*IA 2013)

Turin, 4-6th Decembre, 2013

Context & Motivation (1/2)

Answer Set Programming (ASP)

- Declarative logic programming paradigm
- Real-world applications: AI, KR&R, ... industry

  → **Strengths:** *language expressivity & effective solvers*

Context & Motivation (1/2)

Answer Set Programming (ASP)

- Declarative logic programming paradigm
- Real-world applications: AI, KR&R, ... industry

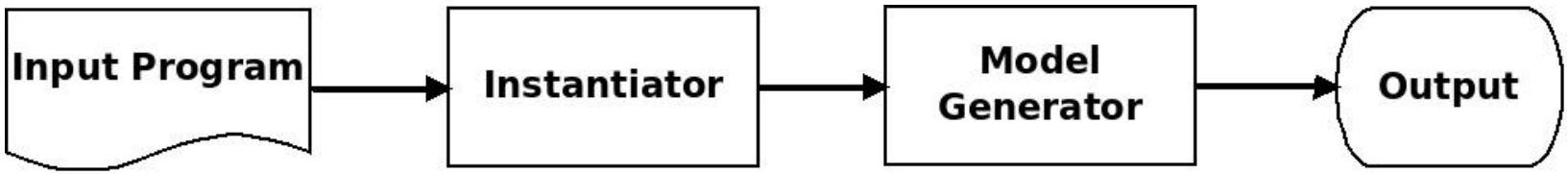
  → **Strengths:** *language expressivity & effective solvers*

The number of ASP applications is further increasing

We need even more efficient ASP systems!

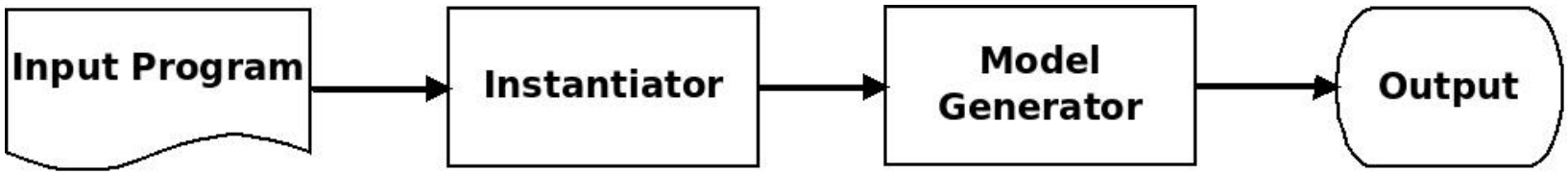
Context & Motivation (2/2)

Evaluation of ASP programs



Context & Motivation (2/2)

Evaluation of ASP programs

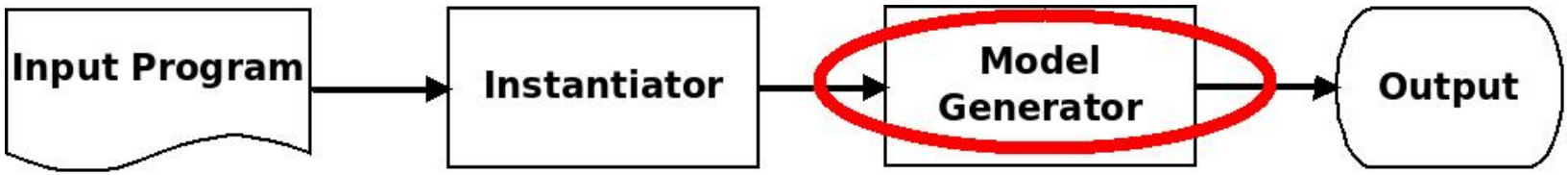


The application of automated **algorithm selection** techniques has improved the performance of ASP systems

→CLASPFOLIO, ME-ASP, **...**

Context & Motivation (2/2)

Evaluation of ASP programs



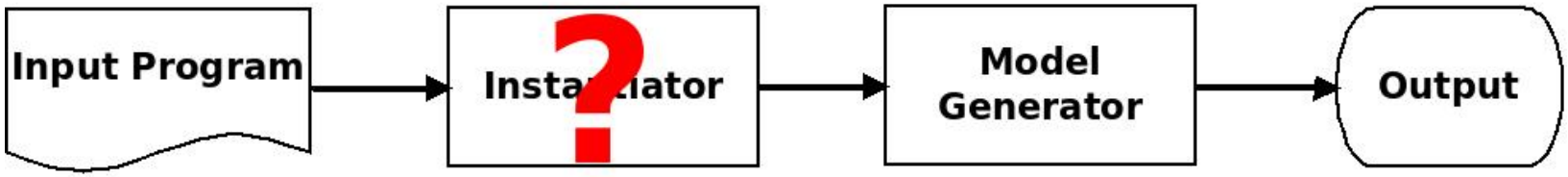
The application of automated **algorithm selection** techniques has improved the performance of ASP systems

$\longrightarrow$CLASPFOLIO, ME-ASP, **...**

Application “confined” to the **Model Generator**

Context & Motivation (2/2)

Evaluation of ASP programs



The application of automated **algorithm selection** techniques has improved the performance of ASP systems

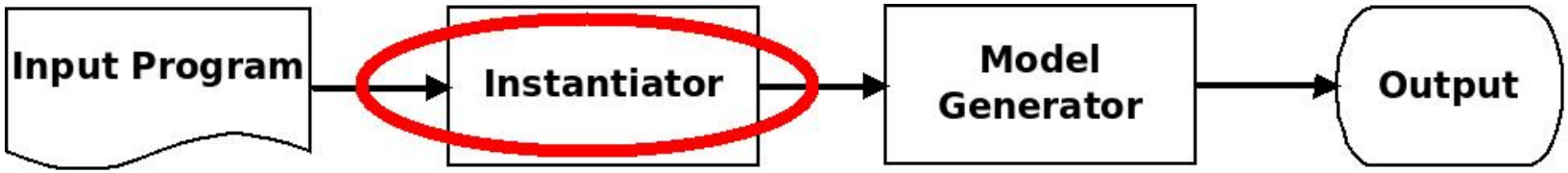
→CLASPFOLIO, ME-ASP, **...**

Application “confined” to the **Model Generator**

What about the **Instantiator**?

Context & Motivation (2/2)

Evaluation of ASP programs



The application of automated **algorithm selection** techniques has improved the performance of ASP systems

→CLASPFOLIO, ME-ASP, **...**

~~Application "confined" to the **Model Generator**~~

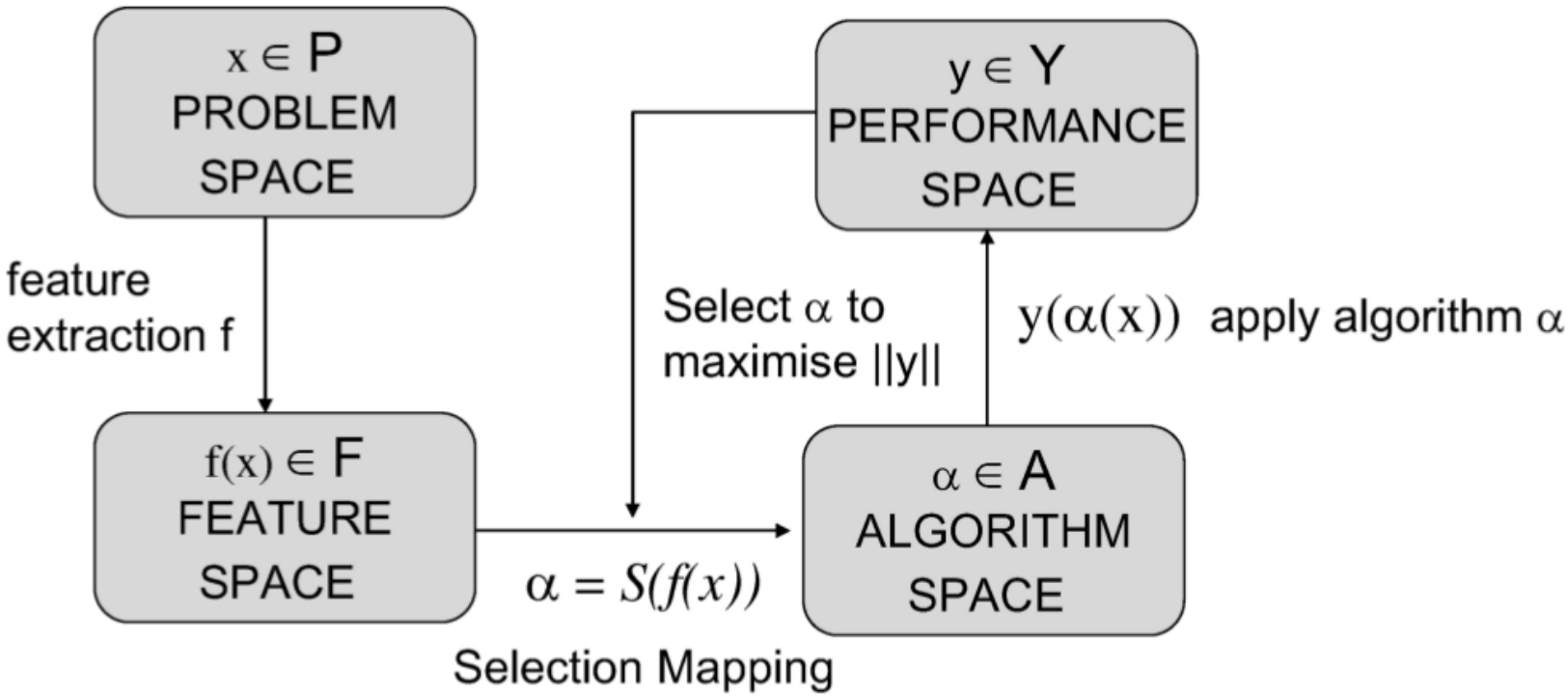
Our contribution

A first step toward the exploitation of automated algorithm selection techniques to the **Instantiator** (or, **grounder).**

Outline

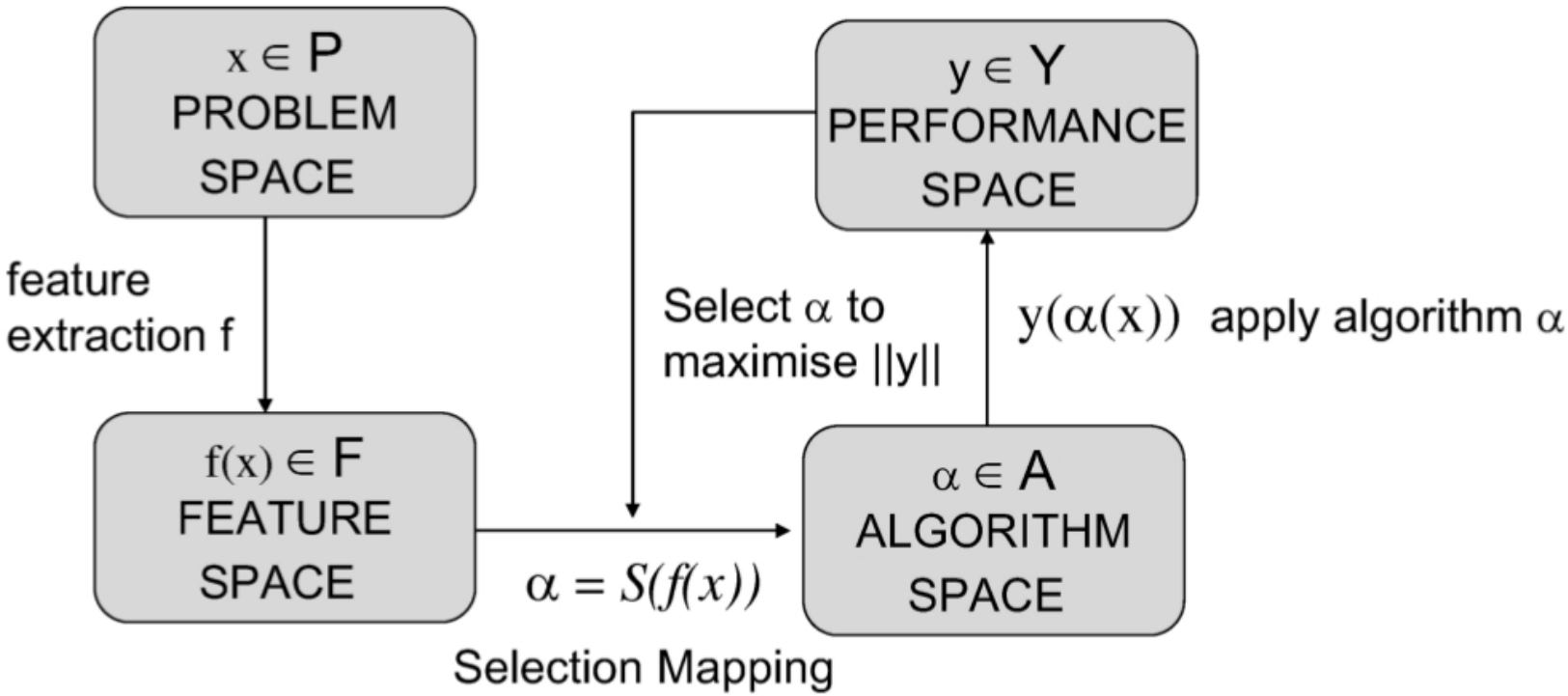
Algorithm Selection Framework

Rice's algorithm selection problem model



Algorithm Selection Framework

Rice's algorithm selection problem model



Objective

Determine the selection mapping *S* that returns the best algorithm α for a problem *x*.

Features and problems

Features:

- Problem size, balance and proximity to Horn features
- Presence of queries
- Maximum Strongly Connected Components size
- Number of Head-Cycle Free (HCF) and non-HCF components
- Features indicating if the program is recursive, tight, stratified
- Number of built-ins
- ...

Features computed on the *P* and *NP* instances **submitted** to the 3rd ASP Competition (**evaluated** instances have been discarded, and are used to test our solution).

Outline

Classification and model

- Too much features to compute a hand-made model
- PART algorithm to automatically build an If-then-else decision list
  - Supervised classification algorithm
  - Patterns are the feature vectors
  - Classes (labels) are the grounders

Classification and model

- Too much features to compute a hand-made model
- PART algorithm to automatically build an If-then-else decision list
  - Supervised classification algorithm
  - Patterns are the feature vectors
  - Classes (labels) are the grounders

Resulting model with DLV-g and GRINGO grounders

- DLV-g is usually preferable when

  $\rightarrow$ One has to deal with queries

  $\rightarrow$ Program contains rules having large bodies (e.g., $\geq$ 4 literals) and the program has a simple structure (few components)
- GRINGO is usually preferable when

  $\rightarrow$ Recursive programs with many components

  $\rightarrow$ Most of the rules have a short body

Tool & Results

Automated grounder SELECTOR

- A feature extractor for non-ground programs
- A decision-making module implementing the model

Tool & Results

Automated grounder SELECTOR

- A feature extractor for non-ground programs
- A decision-making module implementing the model

Solver	Grounder	*P*		*NP*		**Total**	
		#	Time	#	Time	#	Time
CLASP	DLV-g	48	128.26	93	62.65	141	84.99
	GRINGO	35	97.21	72	32.69	107	53.80
	SELECTOR	48	70.94	95	59.64	143	63.43
CMODELS	DLV-g	46	130.47	86	82.42	132	99.16
	GRINGO	32	116.29	67	58.44	99	77.14
	SELECTOR	46	70.60	87	80.72	133	77.22
DLV	DLV-g	41	129.17	59	71.39	100	95.08
	GRINGO	31	107.89	37	28.53	68	64.71
	SELECTOR	41	71.26	59	69.50	100	70.22
IDP	DLV-g	43	136.54	92	71.13	135	91.96
	GRINGO	32	140.57	72	46.97	104	75.77
	SELECTOR	43	74.16	94	70.19	137	71.43

Outline

Conclusion & Future Work

Both grounders and solvers are crucial for the performance of an ASP system

- a first step toward the exploitation of automated selection techniques to the grounding component.

Our grounder selector improves the evaluation performance **independently from the solver** associated.

The grounder selector is available for download at

`http://www.mat.unical.it/ricca/downloads/GR-SELECTOR-AIIA.zip`

Future Work

Selector able to predict the best grounder+solver pair among a set of possible combinations.