Faculty of Mathematics, Physics and Informatics
Comenius University, Bratislava

# Transformational semantics for Evolving Logic Programs

Martin Slota

Supervisors: Ján Šefránek, João Alexandre Leite

March 30, 2007

# Outline

1. Introduction
   - Logic programming
   - Evolving logic programs

2. Transformational semantics for EVOLP

# Introduction

1. Introduction
   - Logic programming
   - Evolving logic programs

2. Transformational semantics for EVOLP

# Logic programming

## Logic program

$$\begin{aligned}
day &\leftarrow \text{not } night.\\
night &\leftarrow \text{not } day.\\
play\_violin &\leftarrow day.\\
play\_piano &\leftarrow night.
\end{aligned}$$

# Logic programming

## Logic program

$$\begin{aligned} day &\leftarrow \text{not } night. \\ night &\leftarrow \text{not } day. \\ play\_violin &\leftarrow day. \\ play\_piano &\leftarrow night. \end{aligned}$$

## Stable models

$$\begin{aligned} M_1 &= \{day, play\_violin\} \\ M_2 &= \{night, play\_piano\} \end{aligned}$$

# Evolving Logic Programs (EVOLP)

- language based on logic programming
- intended for dynamic environments (e.g. multiagent systems)
- syntax is the same as the syntax of logic programs

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} write\_thesis & \leftarrow & \text{not } tired. \\ drink\_coffee & \leftarrow & tired, \text{not } no\_coffee. \\ buy\_coffee & \leftarrow & tired, no\_coffee. \\ assert(tired) & \leftarrow & write\_thesis. \\ assert(\text{not } tired) & \leftarrow & drink\_coffee. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | $\emptyset$ | |
| 3 | | | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | $\emptyset$ | $\{\mathit{tired}, \mathit{drink\_coffee}, \mathit{assert}(\text{not } \mathit{tired})\}$ |
| 3 | | | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | $\emptyset$ | $\{\mathit{tired}, \mathit{drink\_coffee}, \mathit{assert}(\text{not } \mathit{tired})\}$ |
| 3 | $\{\text{not } \mathit{tired}.\}$ | | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} write\_thesis & \leftarrow & \text{not } tired. \\ drink\_coffee & \leftarrow & tired, \text{not } no\_coffee. \\ buy\_coffee & \leftarrow & tired, no\_coffee. \\ assert(tired) & \leftarrow & write\_thesis. \\ assert(\text{not } tired) & \leftarrow & drink\_coffee. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{write\_thesis, assert(tired)\}$ |
| 2 | $\{tired.\}$ | $\emptyset$ | $\{tired, drink\_coffee, assert(\text{not } tired)\}$ |
| 3 | $\{\text{not } tired.\}$ | $\{no\_coffee\}$ | |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | $\emptyset$ | $\{\mathit{tired}, \mathit{drink\_coffee}, \mathit{assert}(\text{not } \mathit{tired})\}$ |
| 3 | $\{\text{not } \mathit{tired}.\}$ | $\{\mathit{no\_coffee}\}$ | $\{\mathit{no\_coffee}, \mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 4 | | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | $\emptyset$ | $\{\mathit{tired}, \mathit{drink\_coffee}, \mathit{assert}(\text{not } \mathit{tired})\}$ |
| 3 | $\{\text{not } \mathit{tired}.\}$ | $\{\mathit{no\_coffee}\}$ | $\{\mathit{no\_coffee}, \mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 4 | $\{\mathit{tired}.\}$ | | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} write\_thesis & \leftarrow & \text{not } tired. \\ drink\_coffee & \leftarrow & tired, \text{not } no\_coffee. \\ buy\_coffee & \leftarrow & tired, no\_coffee. \\ assert(tired) & \leftarrow & write\_thesis. \\ assert(\text{not } tired) & \leftarrow & drink\_coffee. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{write\_thesis, assert(tired)\}$ |
| 2 | $\{tired.\}$ | $\emptyset$ | $\{tired, drink\_coffee, assert(\text{not } tired)\}$ |
| 3 | $\{\text{not } tired.\}$ | $\{no\_coffee\}$ | $\{no\_coffee, write\_thesis, assert(tired)\}$ |
| 4 | $\{tired.\}$ | $\{no\_coffee\}$ | |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} write\_thesis & \leftarrow & \text{not } tired. \\ drink\_coffee & \leftarrow & tired, \text{not } no\_coffee. \\ buy\_coffee & \leftarrow & tired, no\_coffee. \\ assert(tired) & \leftarrow & write\_thesis. \\ assert(\text{not } tired) & \leftarrow & drink\_coffee. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{write\_thesis, assert(tired)\}$ |
| 2 | $\{tired.\}$ | $\emptyset$ | $\{tired, drink\_coffee, assert(\text{not } tired)\}$ |
| 3 | $\{\text{not } tired.\}$ | $\{no\_coffee\}$ | $\{no\_coffee, write\_thesis, assert(tired)\}$ |
| 4 | $\{tired.\}$ | $\{no\_coffee\}$ | $\{tired, no\_coffee, buy\_coffee\}$ |
| 5 | | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | $\emptyset$ | $\{\mathit{tired}, \mathit{drink\_coffee}, \mathit{assert}(\text{not } \mathit{tired})\}$ |
| 3 | $\{\text{not } \mathit{tired}.\}$ | $\{\mathit{no\_coffee}\}$ | $\{\mathit{no\_coffee}, \mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 4 | $\{\mathit{tired}.\}$ | $\{\mathit{no\_coffee}\}$ | $\{\mathit{tired}, \mathit{no\_coffee}, \mathit{buy\_coffee}\}$ |
| 5 | $\emptyset$ | | |

# EVOLP – Example

$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | $\emptyset$ | $\{\mathit{tired}, \mathit{drink\_coffee}, \mathit{assert}(\text{not } \mathit{tired})\}$ |
| 3 | $\{\text{not } \mathit{tired}.\}$ | $\{\mathit{no\_coffee}\}$ | $\{\mathit{no\_coffee}, \mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 4 | $\{\mathit{tired}.\}$ | $\{\mathit{no\_coffee}\}$ | $\{\mathit{tired}, \mathit{no\_coffee}, \mathit{buy\_coffee}\}$ |
| 5 | $\emptyset$ | $\emptyset$ | |

# EVOLP – Example

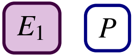
$$P = \left\{ \begin{array}{rcl} \mathit{write\_thesis} & \leftarrow & \text{not } \mathit{tired}. \\ \mathit{drink\_coffee} & \leftarrow & \mathit{tired}, \text{not } \mathit{no\_coffee}. \\ \mathit{buy\_coffee} & \leftarrow & \mathit{tired}, \mathit{no\_coffee}. \\ \mathit{assert}(\mathit{tired}) & \leftarrow & \mathit{write\_thesis}. \\ \mathit{assert}(\text{not } \mathit{tired}) & \leftarrow & \mathit{drink\_coffee}. \end{array} \right\}$$

| Time | Program | Event | Model |
|---|---|---|---|
| 1 | $P$ | $\emptyset$ | $\{\mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 2 | $\{\mathit{tired}.\}$ | $\emptyset$ | $\{\mathit{tired}, \mathit{drink\_coffee}, \mathit{assert}(\text{not } \mathit{tired})\}$ |
| 3 | $\{\text{not } \mathit{tired}.\}$ | $\{\mathit{no\_coffee}\}$ | $\{\mathit{no\_coffee}, \mathit{write\_thesis}, \mathit{assert}(\mathit{tired})\}$ |
| 4 | $\{\mathit{tired}.\}$ | $\{\mathit{no\_coffee}\}$ | $\{\mathit{tired}, \mathit{no\_coffee}, \mathit{buy\_coffee}\}$ |
| 5 | $\emptyset$ | $\emptyset$ | $\{\mathit{tired}, \mathit{drink\_coffee}, \mathit{assert}(\text{not } \mathit{tired})\}$ |

# EVOLP – How it works

$\boxed{P}$

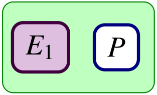
# EVOLP – How it works



$E_1$

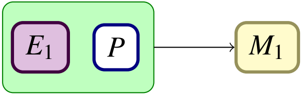
$P$

# EVOLP – How it works

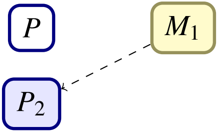


$E_1$ $P$

# EVOLP – How it works

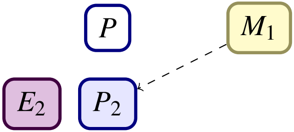


# EVOLP – How it works



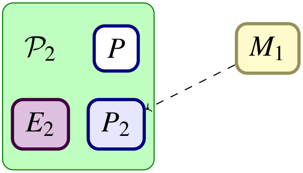
$$P_2 = \{r \mid assert(r) \in M_1\}$$

# EVOLP – How it works



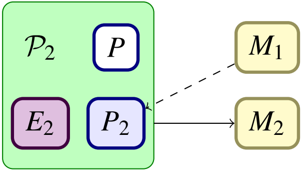
$$P_2 = \{r \mid assert(r) \in M_1\}$$

# EVOLP – How it works



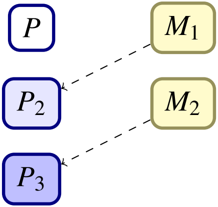
$$P_2 = \{r \mid assert(r) \in M_1\}$$

# EVOLP – How it works



$$P_2 = \{r \mid assert(r) \in M_1\}$$

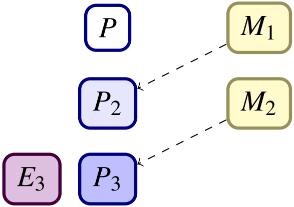
# EVOLP – How it works



$$P_2 = \{r \mid assert(r) \in M_1\}$$

$$P_3 = \{r \mid assert(r) \in M_2\}$$

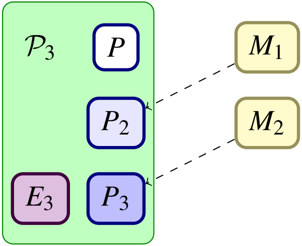
# EVOLP – How it works



$P_2 = \{r \mid assert(r) \in M_1\}$

$P_3 = \{r \mid assert(r) \in M_2\}$

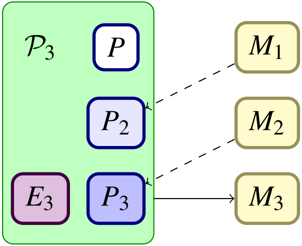
# EVOLP – How it works



$P_2 = \{r \mid assert(r) \in M_1\}$

$P_3 = \{r \mid assert(r) \in M_2\}$

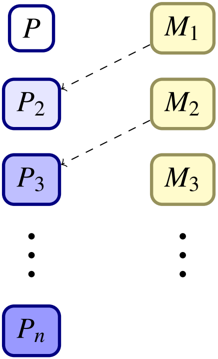
# EVOLP – How it works



$$P_2 = \{r \mid assert(r) \in M_1\}$$

$$P_3 = \{r \mid assert(r) \in M_2\}$$

# EVOLP – How it works



$P$

$M_1$

$P_2$

$M_2$

$P_3$

$M_3$

$\vdots$

$\vdots$

$P_n$

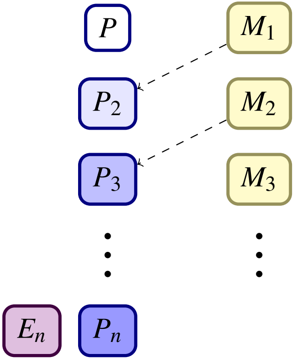
$$P_2 = \{r \mid assert(r) \in M_1\}$$

$$P_3 = \{r \mid assert(r) \in M_2\}$$

$$\vdots$$

$$P_n = \{r \mid assert(r) \in M_{n-1}\}$$

# EVOLP – How it works



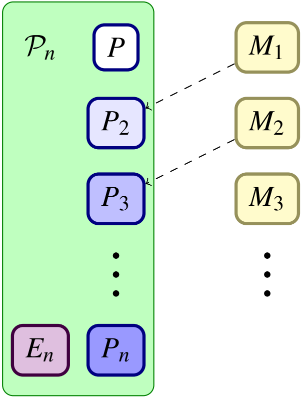
$$P_2 = \{r \mid assert(r) \in M_1\}$$

$$P_3 = \{r \mid assert(r) \in M_2\}$$

$$\vdots$$

$$P_n = \{r \mid assert(r) \in M_{n-1}\}$$

# EVOLP – How it works



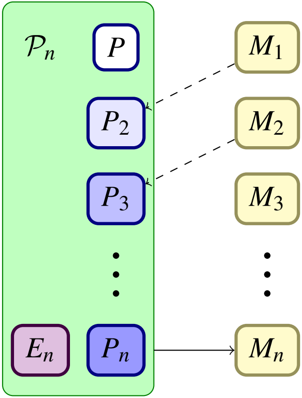
$$P_2 = \{r \mid assert(r) \in M_1\}$$

$$P_3 = \{r \mid assert(r) \in M_2\}$$

$$\vdots$$

$$P_n = \{r \mid assert(r) \in M_{n-1}\}$$

# EVOLP – How it works



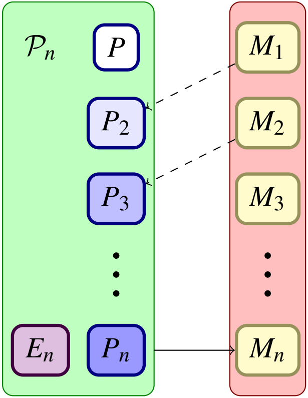
$$P_2 = \{r \mid assert(r) \in M_1\}$$

$$P_3 = \{r \mid assert(r) \in M_2\}$$

$$\vdots$$

$$P_n = \{r \mid assert(r) \in M_{n-1}\}$$

# EVOLP – How it works



Evolution stable model

$P_2 = \{r \mid assert(r) \in M_1\}$

$P_3 = \{r \mid assert(r) \in M_2\}$

$\vdots$

$P_n = \{r \mid assert(r) \in M_{n-1}\}$

# Transformational semantics for EVOLP

1. Introduction
   - Logic programming
   - Evolving logic programs
2. Transformational semantics for EVOLP

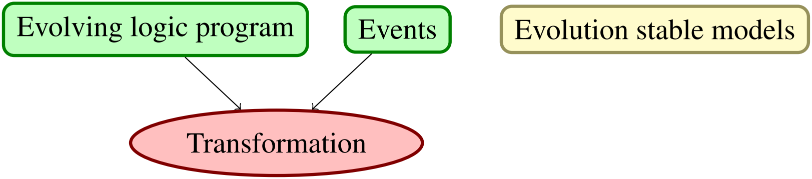
# Transformational semantics for EVOLP

Evolving logic program

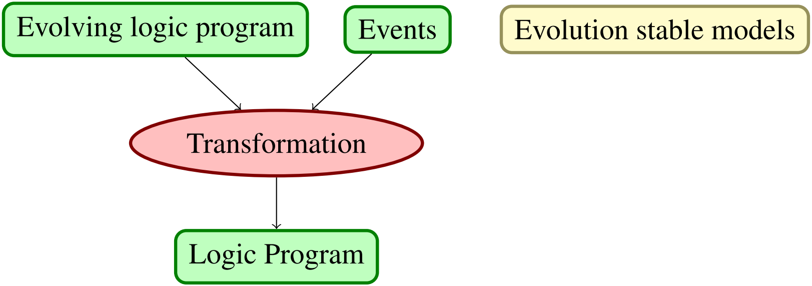
Events

Evolution stable models

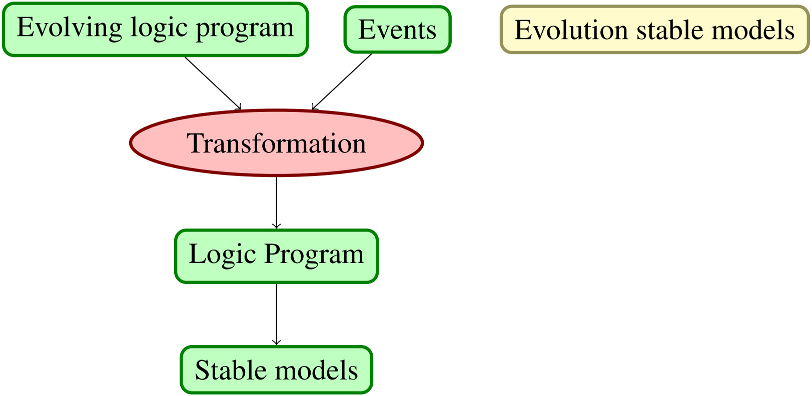
# Transformational semantics for EVOLP



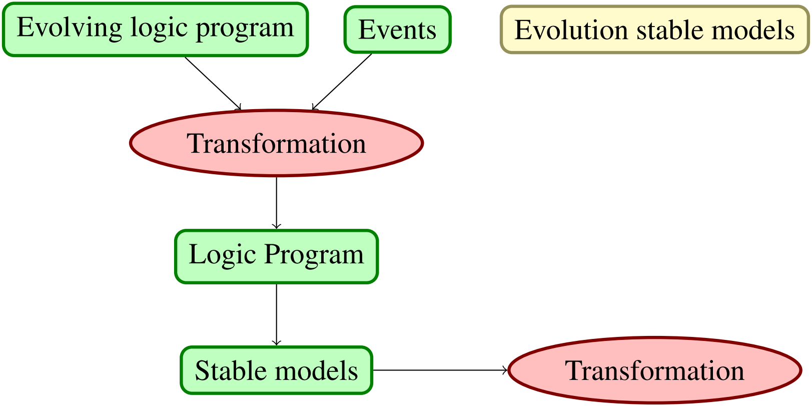
# Transformational semantics for EVOLP



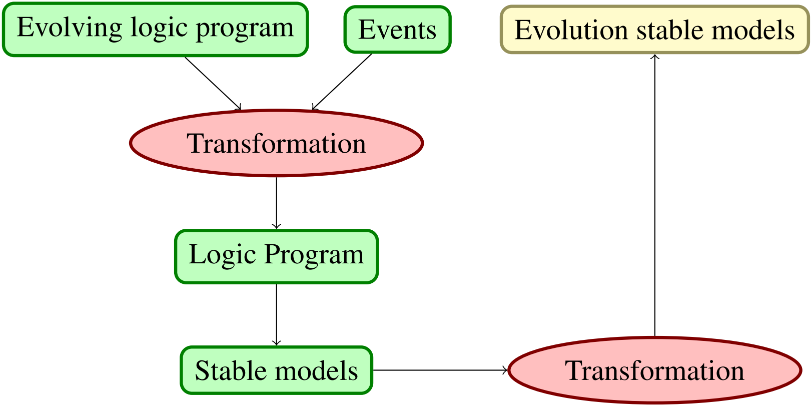
# Transformational semantics for EVOLP



# Transformational semantics for EVOLP



# Transformational semantics for EVOLP



# What is all this good for?

- shows that EVOLP can be, in fact, translated into traditional logic programming (and how)
- shows its semantics from a different perspective
- provides a simple, straight-foward way to implement it

# Results of the thesis

- definition of the transformation
- proofs of its correctness and completeness
- an implementation of propositional EVOLP

# How big is the transformed program?

For an input program $P$ and an event sequence $\mathcal{E} = (E_1, E_2, \ldots, E_n)$ over the universe $\mathcal{U}$ we have:

- transformed program is built of $2n|\mathcal{U}|$ atoms (at most)
- lower bound for the size of the transformed program:

$$|P_{\mathcal{E}}| \geq n|P| + \sum_{k=1}^{n} |E_k|$$

# How big is the transformed program?

For an input program $P$ and an event sequence $\mathcal{E} = (E_1, E_2, \ldots, E_n)$ over the universe $\mathcal{U}$ we have:

- transformed program is built of $2n|\mathcal{U}|$ atoms (at most)
- lower bound for the size of the transformed program:

$$|P_{\mathcal{E}}| \geq n|P| + \sum_{k=1}^{n} |E_k|$$

- upper bound for the size of the transformed program:

$$|P_{\mathcal{E}}| \leq \frac{7}{2}\left(|P|\frac{n^3+5n}{6} + \sum_{k=1}^{n} |E_k| \left(\frac{(n-k)^3+5(n-k)}{6} + 1\right)\right) + n|\mathcal{U}|$$

# How big is the transformed program?

For an input program $P$ and an event sequence $\mathcal{E} = (E_1, E_2, \ldots, E_n)$ over the universe $\mathcal{U}$ we have:

- transformed program is built of $2n|\mathcal{U}|$ atoms (at most)
- lower bound for the size of the transformed program:

$$|P_{\mathcal{E}}| \geq n|P| + \sum_{k=1}^{n} |E_k|$$

- upper bound for the size of the transformed program:

$$|P_{\mathcal{E}}| \leq \frac{7}{2}\left(|P|\frac{n^3+5n}{6} + \sum_{k=1}^{n} |E_k| \left(\frac{(n-k)^3+5(n-k)}{6} + 1\right)\right) + n|\mathcal{U}|$$

- upper bound if input programs contain no nested asserts:

$$|P_{\mathcal{E}}| \leq \frac{7}{2}\left(|P|\frac{n^2+n}{2} + \sum_{k=1}^{n} |E_k|(n-k+1)\right) + n|\mathcal{U}|$$

# Future work

- extensions of the existing implementation:
  - variable support
  - support for arithmetic predicates
  - ...
- a different, more direct implementation that could be used on-line

# Thank you

Thank you.
Are there any questions?