Making Universal Induction Efficient by Specialization

AGI @ Quebec

Alexey Potapov, Sergey Rodionov

{potapov, rodionov}@aideus.com

2014

General Intelligence

(General) intelligence is an agent's ability to efficiently achieve goals in a wide range of environments with insufficient knowledge and resources.

Gap between Universal and Pragmatic Methods

- Universal methods
  - can work in arbitrary computable environment
  - computationally infeasible
  - approximations are either inefficient or not universal
- Pragmatic methods
  - work in non-toy environments
  - set of environments is highly restricted

=> Bridging this gap is necessary

Key Idea

- Humans create narrow methods, which efficiently solve arbitrary recurring problems
- Generality should be achieved not by a single uniform method solving any problem in the same fashion, but by automatic construction of (non-universal) efficient methods
- Program specialization is the appropriate concept*, which relates general and narrow intelligence methods
- However, no analysis of possible specialization of concrete models of universal intelligence has been given yet.

* Khudobakhshov, V.: Metacomputations and Program-based Knowledge Representation. In:K.-U. Kühnberger, S. Rudolph, P. Wang (Eds.): AGI'13, LNAI 7999, pp. 70–77 (2013).

Program Specialization

- Let $p_L(x,y)$ be some program (in some language L) with two arguments
- Specializer $spec_R$ is such program (in some language R) accepting p_L and x_0 that

$$(\forall y) spec_R(p_L, x_0)(y) = p_L(x_0, y)$$

- $spec_R(p_L, x_0)$ is the result of deep transformation of p_L that can be much more efficient than $p(x_0, .)$

Futamura-Turchin projections

$$(\forall x) spec_R(intL, p_L)(x) = intL(p_L, x)$$

$$(\forall p_L, x) spec_R(spec_R, intL)(p_L)(x) = intL(p_L, x)$$

$$(\forall intL) spec_R(spec_R, spec_R)(intL) = comp_{L \to R}$$

Universal Mass Induction

- Let $\{x_i\}_{i=1}^{n}$ be the set of strings
- An universal method cannot be applied to mass problems since typically

$$K_U(x_1 x_2 \ldots x_n) << \sum_{i=1}^{n} K_U(x_i)$$

where *K* is Kolmogorov complexity on universal machine *U*

- However, $K_U(x_1 x_2 \ldots x_n) \approx \min_S \left(l(S) + \sum_{i=1}^{n} K_U(x_i \mid S) \right)$ can be true
- One can search for models $y_i^* = \arg\min_{y:S(y)=x_i} l(y)$ for each x_i independently

within some best representation $S^* = \arg\min_S \left(l(S) + \sum_{i=1}^{n} l(y_i^*) \right)$

If *S* is not an universal program than this search can be made (much) more efficient than exhaustive search

Specialization of Universal Induction

- Universal mass induction consists of two procedures
  - Search for models

$$MSearch(S, x_i) \to y_i^* = \arg\min_{y:S(y)=x_i} l(y)$$

  - Search for representations

$$RSearch(x_1,\ldots x_n) \to S^* = \arg\min_{S}\left(l(S) + \sum_{i=1}^{n} l(y_i^*) \right)$$

- *MSearch*(*S*, *x*) is executed for different *x* with same *S*
- This search cannot be non-exhaustive for any *S*, but it can be efficient for some of them
- One can consider computationally efficient projection *spec*(*MSearch*, *S*): $(\forall x)spec(MSearch, S)(x) = MSearch(S, x)$

Approach to Specialization

- Direct specialization of *MSearch*(S, x) w.r.t. some given S^*
  - No general techniques for exponential speedup exists
  - And how to get S? *RSearch* is still needed
- Find $S'=spec(MSearch(S, x), S^*)$ simultaneously with S^*
  - Main properties of S, S': $(\forall x)S(S'(x)) = x$

$$l(S) + \sum_i l(S'(x_i)) \to \min$$

- S is a generative representation (decoding)
- S' is a descriptive representation (encoding)
  - S' is also the result of specialization of the search for generative models, so in general it can include some sort of optimized search
- Simultaneous search for S and S' will be referred to as *SS'-search*

Combinatory Logic

- $\mathbf{K}\ x\ y \rightarrow x$ $\qquad ((\mathbf{K}\ x)\ y)$
- $\mathbf{S}\ x\ y\ z \rightarrow x\ z\ (y\ z)$ $\qquad (((\mathbf{S}\ x)\ y)\ z)$
  - $\mathbf{S}\ \mathbf{K}\ \mathbf{K}\ x \rightarrow \mathbf{K}\ x\ (\mathbf{K}\ x) \rightarrow x$ $\qquad \mathbf{I} = \mathbf{S}\ \mathbf{K}\ \mathbf{K}$ $\qquad \mathbf{I}\ x \rightarrow x$
  - $(\mathbf{S}\ (\mathbf{K}\ (\mathbf{S}\ \mathbf{I}))\ (\mathbf{S}\ (\mathbf{K}\ \mathbf{K})\ \mathbf{I})\ x\ y) \rightarrow \ldots \rightarrow y\ x$
  - and other combinators: **B**, **b**, **W**, **M**, **J**, **C**, **T**
- In lambda-calculus
  - $\lambda x.x == \mathbf{I}$ $\qquad \lambda x.\ \lambda y.(y\ x) == \mathbf{S}\ (\mathbf{K}\ (\mathbf{S}\ \mathbf{I}))\ (\mathbf{S}\ (\mathbf{K}\ \mathbf{K})\ \mathbf{I})$

Mass Induction in CL

- 0 1 0 2 ← S 0 1 2
- 3 0 3 1 ← S 3 0 1
- 2 1 2 0 ← S 2 1 0

Individual models y_i

One representation S

Data strings x_i with common regularities

- *MSearch* enumerates all models to find the shortest appropriate model: $Sy_i=x_i$
- *RSearch* enumerates all S and calls *MSearch* for each S

SS'-Search example

$S' = \mathbf{KC}$

- 0 1 0 2 ← **S** 0 1 2
- 3 0 3 1 ← **S** 3 0 1
- 2 1 2 0 ← **S** 2 1 0

Individual models y_i

One representation S

Data strings x_i with common regularities

- S and S' are enumerated together
- S' is used instead of *MSearch* to obtain y_i

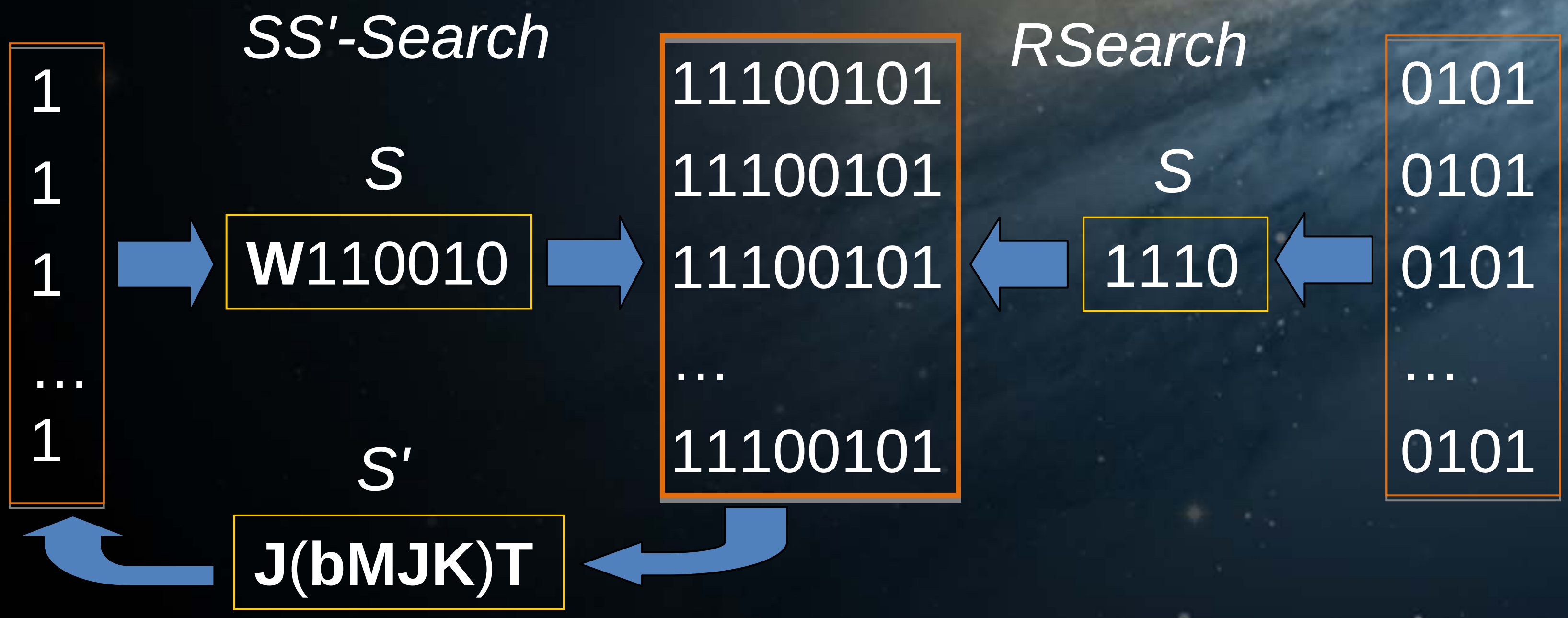
Genetic programming for Mass Induction

- *RSearch+MSearch*
  - Genome is composed of S and $\{y_i\}$ each of which corresponds to a separate chromosome
- *SS'-Search*
  - Genome is composed of two chromosomes – S and S'
- Each chromosome is subjected to crossover independently
- Implementation of GP for CL is described in our previous paper*

* Potapov, A., Rodionov, S.: Universal Induction with Varying Sets of Combinators. In: K.-W. Kühnberger, S. Rudolph, P. Wang (Eds.): AGI'13, LNAI 7999, pp. 88–97 (2013).

Experimental results

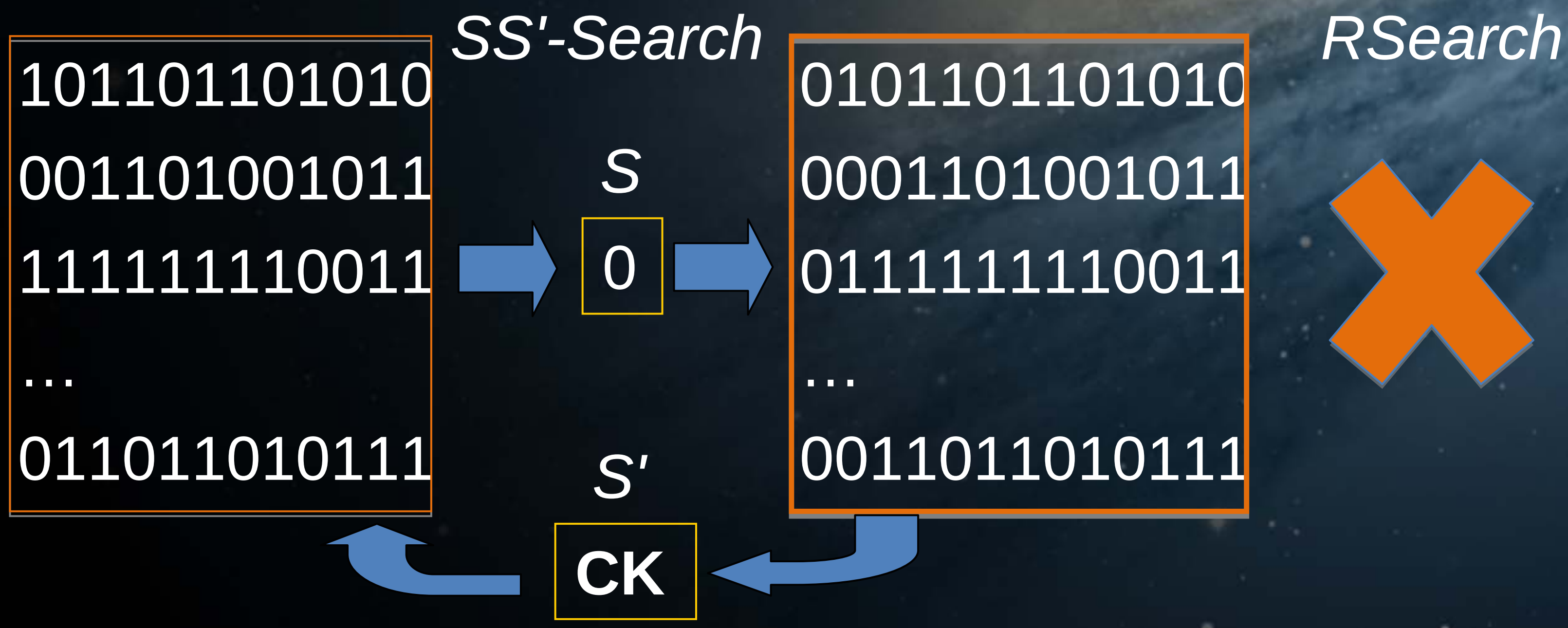
- Simple redundancy



- *RSearch* fails to find optimal solution even in this simple case
- *SS'-Search* appears to be efficient; *S'* constructs correct models
- This can seem strange since *S'* is not simpler than y_i, but *SS'-Search* allows for incremental improvement

Experimental results

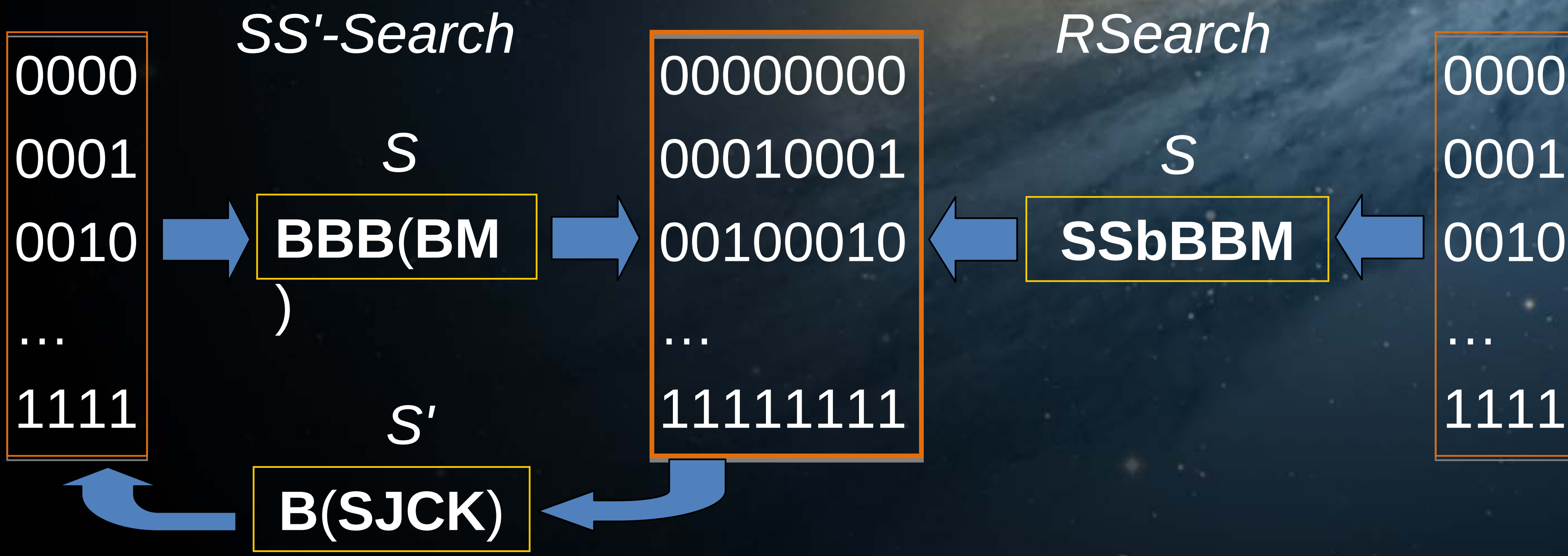
- Poorly compressible data



- *RSearch* fails to find any precise solution
- *SS'-Search* extracts information from data to construct models, while *RSearch* searches for models blindly

Experimental results

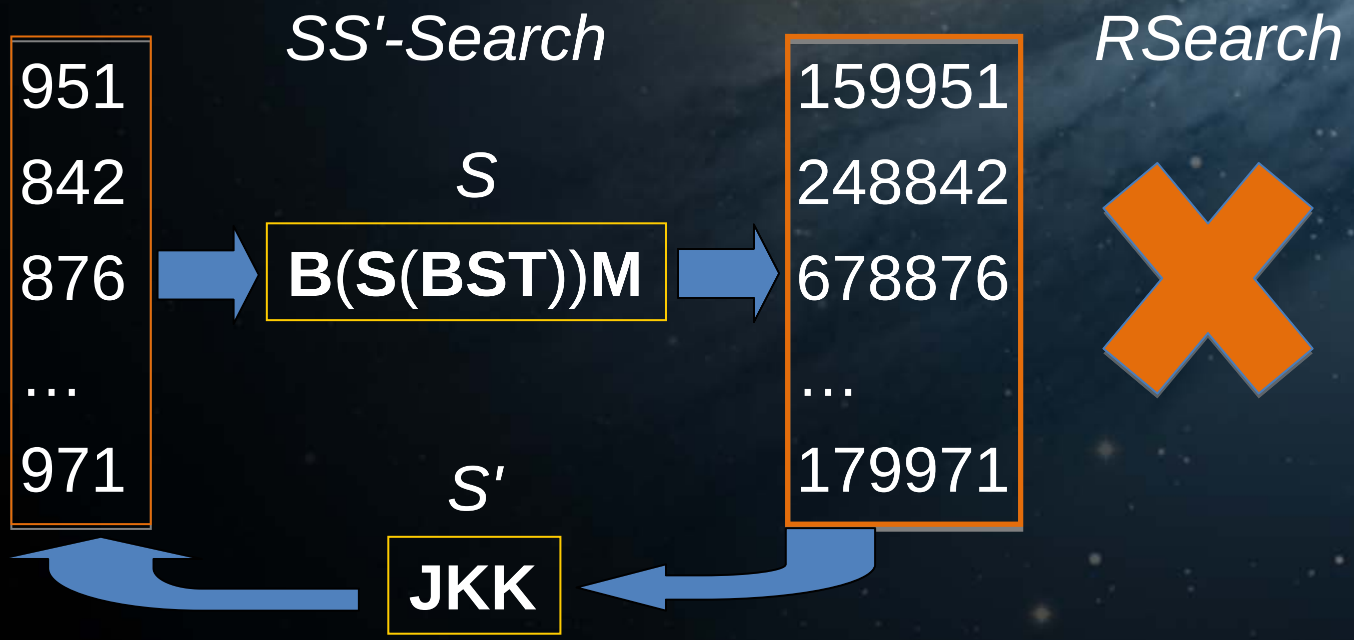
- Simple common regularity



- Both methods successfully found good solutions
- *RSearch* requires low complexity from both representations and models

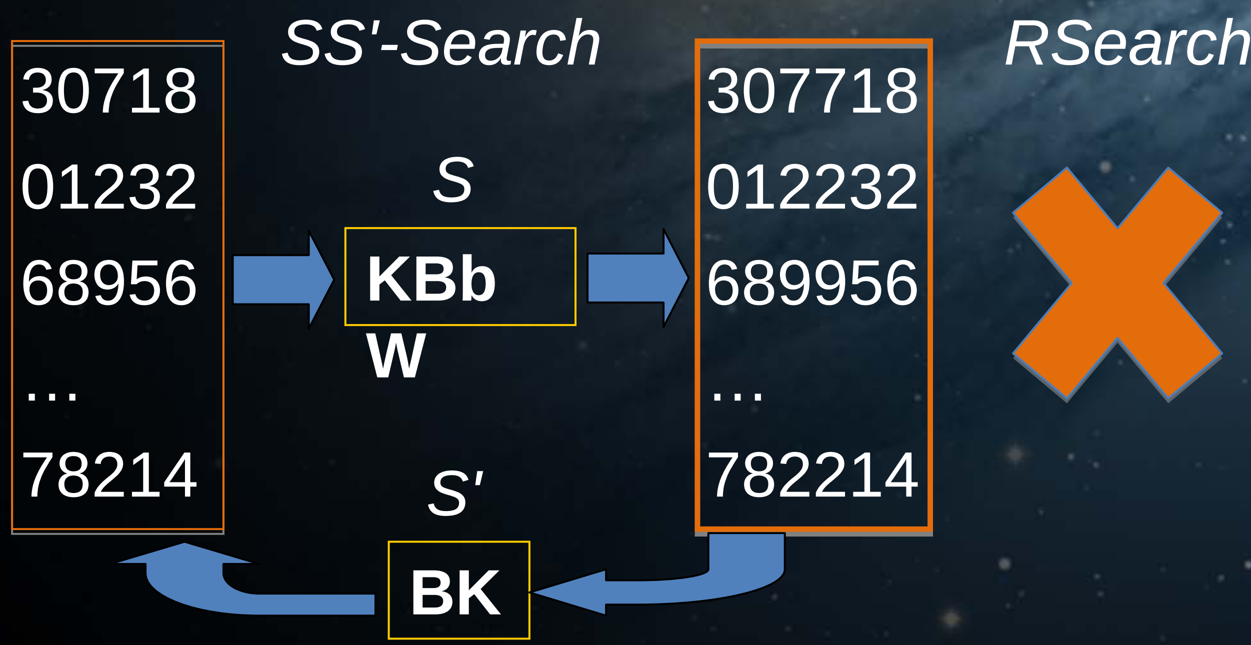
Experimental results

- More complex regularities



Experimental results

- More complex regularities



Conclusion

- Ideas of universal induction, representations, and program specialization were combined
- Specialization of universal (mass) induction w.r.t. some (generative) representation yields descriptive representations.
- These descriptive representations being not Turing-complete can construct data models much more efficient than universal induction methods
- Also, automatic simultaneous construction of generative and descriptive representations appeared to be more efficient than construction of generative representations and models, so explicit specialization seems to be not necessary here.
- Can *RSearch* be more efficient than *SS'-Search*?

Thank you for attention

AGI @ Quebec

Alexey Potapov, Sergey Rodionov
{potapov, rodionov}@aideus.com

2014