Factored Mappings as Knowledge Compilation for Symbolic Search

Leonhard Badenberg <leonhard.badenberg@stud.unibas.ch>

Department of Mathematics and Computer Science, University of Basel

June 14, 2021

Motivation

- Symbolic Search
  - Approach to classical planning
  - Processes sets of states at a time
- Knowledge Compilations
  - Represent knowledge bases as compact data structure
  - Used for symbolic search (e.g. BDDs)
- Factored Mappings
  - Also called merge-and-shrink representations
  - Represent functions that map variable assignments to a set of values

Motivation

- Symbolic Search
  - Approach to classical planning
  - Processes sets of states at a time
- Knowledge Compilations
  - Represent knowledge bases as compact data structure
  - Used for symbolic search (e.g. BDDs)
- Factored Mappings
  - Also called merge-and-shrink representations
  - Represent functions that map variable assignments to a set of values

Motivation

- Symbolic Search
  - Approach to classical planning
  - Processes sets of states at a time
- Knowledge Compilations
  - Represent knowledge bases as compact data structure
  - Used for symbolic search (e.g. BDDs)
- Factored Mappings
  - Also called merge-and-shrink representations
  - Represent functions that map variable assignments to a set of values

Motivation

- Symbolic Search
  - Approach to classical planning
  - Processes sets of states at a time
- Knowledge Compilations
  - Represent knowledge bases as compact data structure
  - Used for symbolic search (e.g. BDDs)
- Factored Mappings
  - Also called merge-and-shrink representations
  - Represent functions that map variable assignments to a set of values

Can we use Factored Mappings as knowledge compilation for symbolic search?

Planning Task

A planning task is given by $\langle V, I, O, \gamma \rangle$ where

- V is a finite set of state variables,
- I is a valuation over V called the initial state,
- O is a finite set of operators over V, and
- γ is a formula over V called the goal.

Propositional: $\alpha : V \to \{\mathbf{T}, \mathbf{F}\}$

Finite-domain representation (FDR): $\alpha : V \to \bigcup_{v \in V} dom(v)$

Factored Mappings

A Factored Mapping (FM) σ over V is either

- atomic with associated variable $v \in V$, or
- a merge of two components σ_L and σ_R

Atomic FM tables: σ_v^{tab} spanned by $dom(v)$

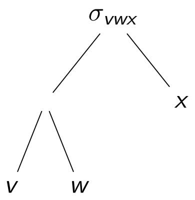
Merge FM tables: σ_m^{tab} spanned by the values of its two components

Component tables σ_i^{tab} are filled with arbitrary different entries

Root table σ_{root}^{tab} is filled with 0 and 1

Can we use FMs for symbolic search on FDR planning tasks?

Factored Mappings



Underlying merge tree $\mathcal{T}(V)$

σ^{tab}_{vwx}	0	1	2
0	0	0	0
1	0	0	1
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0

σ^{tab}_{vw}	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

σ^{tab}_{x}	0	1	2
-	0	1	2

σ^{tab}_{v}	0	1	2
-	0	1	2

σ^{tab}_{w}	0	1	2
-	0	1	2

$A(\sigma) = \{\alpha\} = \{\{v \mapsto 0, w \mapsto 1, x \mapsto 2\}\}$

Requirements

1. FMs over V have the same underlying merge tree $\mathcal{T}(V)$
2. Component tables σ_i^{tab} are filled with fixed value order $0, 1, \ldots, n-1$

This leads to $A(\sigma) = A(\gamma) \iff \sigma = \gamma$.

Reducing FMs

σ^{tab}_{vwx}	0	1	2
0	0	0	0
1	0	0	1
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0

σ^{tab}_{vw}	0	1	2
0	0	1	2
1	3	4	5
2	6	7	8

σ^{tab}_{x}	0	1	2
-	0	1	2

σ^{tab}_{v}	0	1	2
-	0	1	2

σ^{tab}_{w}	0	1	2
-	0	1	2

$$A(\sigma) = \{\alpha\} = \{\{v \mapsto 0, w \mapsto 1, x \mapsto 2\}\}$$

Reducing FMs

σ_{vwx}^{tab}	0	1	2
0	0	0	0
1	0	0	1
~~2~~	~~0~~	~~0~~	~~0~~
~~3~~	~~0~~	~~0~~	~~0~~
~~4~~	~~0~~	~~0~~	~~0~~
~~5~~	~~0~~	~~0~~	~~0~~
~~6~~	~~0~~	~~0~~	~~0~~
~~7~~	~~0~~	~~0~~	~~0~~
~~8~~	~~0~~	~~0~~	~~0~~

σ_{vw}^{tab}	0	1	2
0	0	1	0
1	0	0	0
2	0	0	0

σ_{x}^{tab}	0	1	2
-	0	1	2

σ_{v}^{tab}	0	1	2
-	0	1	2

σ_{w}^{tab}	0	1	2
-	0	1	2

$$A(\sigma) = \{\alpha\} = \{\{v \mapsto 0, w \mapsto 1, x \mapsto 2\}\}$$

Reducing FMs

σ^{tab}_{vwx}	0	~~1~~	1
0	0	~~0~~	0
1	0	~~0~~	1
~~2~~	~~0~~	~~0~~	~~0~~
~~3~~	~~0~~	~~0~~	~~0~~
~~4~~	~~0~~	~~0~~	~~0~~
~~5~~	~~0~~	~~0~~	~~0~~
~~6~~	~~0~~	~~0~~	~~0~~
~~7~~	~~0~~	~~0~~	~~0~~
~~8~~	~~0~~	~~0~~	~~0~~

σ^{tab}_{vw}	0	1	2
0	0	1	0
1	0	0	0
2	0	0	0

σ^{tab}_{x}	0	1	2
-	0	0	1

σ^{tab}_{v}	0	1	2
-	0	1	2

σ^{tab}_{w}	0	1	2
-	0	1	2

$$A(\sigma) = \{\alpha\} = \{\{v \mapsto 0, w \mapsto 1, x \mapsto 2\}\}$$

Reducing FMs

σ_{vwx}^{tab}	0	~~1~~	1
0	0	~~0~~	0
1	0	~~0~~	1
~~2~~	~~0~~	~~0~~	~~0~~
~~3~~	~~0~~	~~0~~	~~0~~
~~4~~	~~0~~	~~0~~	~~0~~
~~5~~	~~0~~	~~0~~	~~0~~
~~6~~	~~0~~	~~0~~	~~0~~
~~7~~	~~0~~	~~0~~	~~0~~
~~8~~	~~0~~	~~0~~	~~0~~

σ_{vw}^{tab}	0	1	~~2~~
0	0	1	~~0~~
1	0	0	~~0~~
~~2~~	~~0~~	~~0~~	~~0~~

σ_{x}^{tab}	0	1	2
-	0	0	1

σ_{v}^{tab}	0	1	2
-	0	1	1

σ_{w}^{tab}	0	1	2
-	0	1	0

$$A(\sigma) = \{\alpha\} = \{\{v \mapsto 0, w \mapsto 1, x \mapsto 2\}\}$$

Reducing FMs

σ^{tab}_{vwx}	0	1
0	0	0
1	0	1

σ^{tab}_{vw}	0	1
0	0	1
1	0	0

σ^{tab}_{x}	0	1	2
-	0	0	1

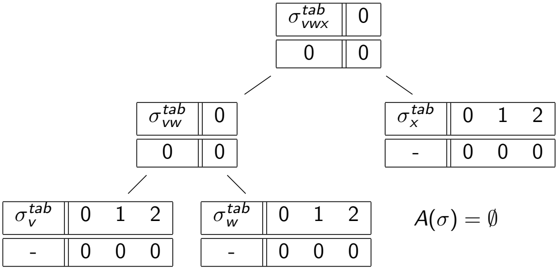
σ^{tab}_{v}	0	1	2
-	0	1	1

σ^{tab}_{w}	0	1	2
-	0	1	0

$$A(\sigma) = \{\alpha\} = \{\{v \mapsto 0, w \mapsto 1, x \mapsto 2\}\}$$

Building Basic Factored Mappings

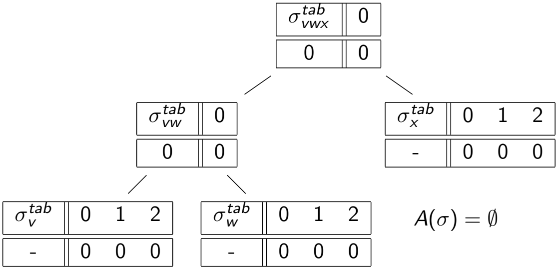
- False ($\perp$): $A(\sigma) = \emptyset$
  - Fills all entries in all tables with zeroes
- True ($\top$): $A(\sigma) = A(V)$
  - Fills all entries in all tables with zeroes
  - Sets entry in root table to 1



- Atom ($v = c$): $A(\sigma) = \{\alpha \mid \alpha[v] = c\}$
  - Fills σ^{tab}_{v} with 0 and 1
  - Fills all other atomic tables with zeroes
  - Keeps different values distinct until the root table

Building Basic Factored Mappings

- False ($\perp$): $A(\sigma) = \emptyset$
  - Fills all entries in all tables with zeroes
- True ($\top$): $A(\sigma) = A(V)$
  - Fills all entries in all tables with zeroes
  - Sets entry in root table to 1
- Atom ($v = c$): $A(\sigma) = \{\alpha \mid \alpha[v] = c\}$
  - Fills σ_v^{tab} with 0 and 1
  - Fills all other atomic tables with zeroes
  - Keeps different values distinct until the root table



Building Basic Factored Mappings

- False ($\bot$): $A(\sigma) = \emptyset$
  - Fills all entries in all tables with zeroes
- True ($\top$): $A(\sigma) = A(V)$
  - Fills all entries in all tables with zeroes
  - Sets entry in root table to 1
- Atom ($v = c$): $A(\sigma) = \{\alpha \mid \alpha[v] = c\}$
  - Fills σ_v^{tab} with 0 and 1
  - Fills all other atomic tables with zeroes
  - Keeps different values distinct until the root table

σ_{vwx}^{tab}	0
0	0
1	1

σ_{vw}^{tab}	0
0	0
1	1

σ_{x}^{tab}	0	1	2
-	0	0	0

σ_{v}^{tab}	0	1	2
-	0	1	0

σ_{w}^{tab}	0	1	2
-	0	0	0

$A(\sigma) = \{\alpha \mid \alpha[v] = 1\}$

For all of these: $\mathcal{O}(n \cdot D^{max})$, where $n = |V|$ and $D^{max} = \max_{v \in V} |dom(v)|$

Boolean Tests

- Includes ($I \models \phi$): Is $\alpha \in A(\sigma)$?
  - Follows the assignment from leaves to root
  - Checks if entry in root node is 1
  - $\mathcal{O}(n)$
- Equals ($\phi \equiv \psi$): Is $A(\sigma) = A(\gamma)$?
  - Because FMs are canonical: $A(\sigma) = A(\gamma) \iff \sigma = \gamma$
  - Checks every entry in σ and γ for equality
  - $\mathcal{O}(n(T_\sigma^{max} + T_\gamma^{max}))$

Where $n = |V|$, and T_σ^{max} and T_γ^{max} is the maximum over all table sizes of σ and γ respectively

Combine

- Combines two FMs σ and γ into one FM δ
- Combines component table pairs σ_i^{tab} and γ_i^{tab} directly into δ_i^{tab}
- δ_i^{tab} is initially filled with 2-dimensional entries

We combine atomic leaf node tables differently than merge node tables

Combining Leaves

σ_l^{tab}	0	1	2	3
-	0	1	0	2

γ_l^{tab}	0	1	2	3
-	0	1	2	0

δ_l^{tab}	0	1	2	3
-	(0,0)	(1,1)	(0,2)	(2,0)

Two leaf node tables σ_l^{tab} and γ_l^{tab} get combined to one δ_l^{tab} with 2-dimensional values.

Combining Merges

σ_m^{tab}	0	1	2
0	0	1	2
1	3	4	5
2	0	2	3

γ_m^{tab}	0	1	2
0	0	1	0
1	0	1	2
2	1	2	3

δ_m^{tab}	(0,0)	(1,1)	(0,2)	(2,0)
(0,0)	(0,0)	(1,1)	(0,0)	(2,0)
(1,1)	(3,0)	(4,1)	(3,2)	(5,0)
(2,2)	(0,1)	(2,2)	(0,3)	(3,1)

$\delta_{m_L}^{tab}$	0	1	2	3
-	(0,0)	(1,1)	(2,2)	(0,0)

$\delta_{m_R}^{tab}$	0	1	2	3
-	(0,0)	(1,1)	(0,2)	(2,0)

Combining Merges

σ_m^{tab}	0	1	2
0	0	1	2
1	3	4	5
2	0	2	3

γ_m^{tab}	0	1	2
0	0	1	0
1	0	1	2
2	1	2	3

δ_m^{tab}	0	1	2	3
0	(0,0)	(1,1)	(0,0)	(2,0)
1	(3,0)	(4,1)	(3,2)	(5,0)
2	(0,1)	(2,2)	(0,3)	(3,1)

$\delta_{m_L}^{tab}$	0	1	2	3
-	0	1	2	0

$\delta_{m_R}^{tab}$	0	1	2	3
-	0	1	2	3

Set Operations

- Union $(\phi \vee \psi)$: $A(\sigma) \cup A(\gamma)$
  - Uses COMBINE to create δ
  - Maps entries (x, y) in δ^{tab}_{root} to 1 if $x = 1$ **or** $x = 1$
  - $\mathcal{O}(n \cdot T^{max}_{\sigma} \cdot T^{max}_{\gamma})$
- Intersection $(\phi \wedge \psi)$: $A(\sigma) \cap A(\gamma)$
  - Uses COMBINE to create δ
  - Maps entries (x, y) in δ^{tab}_{root} to 1 if $x = 1$ **and** $x = 1$
  - $\mathcal{O}(n \cdot T^{max}_{\sigma} \cdot T^{max}_{\gamma})$
- Complement $(\neg\phi)$: $A(\overline{\sigma}) = \overline{A(\sigma)}$
  - Swaps all zeroes and ones in the root table
  - $\mathcal{O}(T^{max}_{\sigma})$

Where $n = |V|$, and T^{max}_{σ} and T^{max}_{γ} is the maximum over all table sizes of σ and γ respectively

Symbolic Search Algorithm

Algorithm 1 Progression Breadth-first Search

```
function BfsProgression(V, I, O, γ)
    goalStates ← Models(γ)
    reached_0 ← {I}
    i ← 0
    loop
        if reached_i ∩ goalStates ≠ 0 then
            return solution found
        reached_{i+1} ← reached_i ∪ Apply(reached_i, O)
        if reached_{i+1} = reached_i then
            return no solution exists
        i ← i + 1
```

Symbolic Search Algorithm

Algorithm 1 Progression Breadth-first Search

function $\textsc{BfsProgression}(V, I, O, \gamma)$
    $goalStates \leftarrow \textsc{Models}(\gamma)$
    $reached_0 \leftarrow \{I\}$
    $i \leftarrow 0$
    loop
        if $reached_i \cap goalStates \neq 0$ **then**
            return solution found
        $reached_{i+1} \leftarrow reached_i \cup \textsc{Apply}(reached_i, O)$
        if $reached_{i+1} = reached_i$ **then**
            return no solution exists
        $i \leftarrow i + 1$

Formula and Singleton

Formula

- Converts formulas ϕ into FMs σ, representing $\textsc{Models}(\phi)$
- Uses the introduced operations and their combinations
- Can take exponentially long

Singleton

- Converts the single assignment I into an FM σ, representing $\{I\}$
- Uses Intersection and Atom

$$\{I\} := \{\{v \mapsto 0, w \mapsto 1, x \mapsto 2\}\} = \{\alpha \mid \alpha[v] = 0\} \cap \{\alpha \mid \alpha[w] = 1\} \cap \{\alpha \mid \alpha[x] = 2\}$$

Symbolic Search Algorithm

Algorithm 1 Progression Breadth-first Search

```
function BfsProgression(V, I, O, γ)
    goalStates ← Models(γ)
    reached_0 ← {I}
    i ← 0
    loop
        if reached_i ∩ goalStates ≠ 0 then
            return solution found
        reached_{i+1} ← reached_i ∪ Apply(reached_i, O)
        if reached_{i+1} = reached_i then
            return no solution exists
        i ← i + 1
```

The Apply function

- Computes the set of states that can be reached by applying operators $o \in O$ in states $s \in \mathit{reached}$
- Stores it as set of assignments inside an FM
- Needs transition relation $T_V(O) = \textsc{Formula}(\bigvee_{o \in O} \tau_V(o))$
- $T_V(O)$ needs variables from V and from V' to describe transitions
- For all FMs σ: σ_L over V and σ_R over V'
- Computes intersection between *reached* and $T_V(O)$
- Reorders and renames the state variables, so the new states over V' will be over V

The Apply function

- Computes the set of states that can be reached by applying operators $o \in O$ in states $s \in \mathit{reached}$
- Stores it as set of assignments inside an FM
- Needs transition relation $T_V(O) = \textsc{Formula}(\bigvee_{o \in O} \tau_V(o))$
- $T_V(O)$ needs variables from V and from V' to describe transitions
- For all FMs σ: σ_L over V and σ_R over V'
- Computes intersection between *reached* and $T_V(O)$
- Reorders and renames the state variables, so the new states over V' will be over V

The Apply function

- Computes the set of states that can be reached by applying operators $o \in O$ in states $s \in \mathit{reached}$
- Stores it as set of assignments inside an FM
- Needs transition relation $T_V(O) = \textsc{Formula}(\bigvee_{o \in O} \tau_V(o))$
- $T_V(O)$ needs variables from V and from V' to describe transitions
- For all FMs σ: σ_L over V and σ_R over V'
- Computes intersection between *reached* and $T_V(O)$
- Reorders and renames the state variables, so the new states over V' will be over V

The Apply function

- Computes the set of states that can be reached by applying operators $o \in O$ in states $s \in \mathit{reached}$
- Stores it as set of assignments inside an FM
- Needs transition relation $T_V(O) = \textsc{Formula}(\bigvee_{o \in O} \tau_V(o))$
- $T_V(O)$ needs variables from V and from V' to describe transitions
- For all FMs σ: σ_L over V and σ_R over V'
- Computes intersection between *reached* and $T_V(O)$
- Reorders and renames the state variables, so the new states over V' will be over V

Time complexity of $\mathcal{O}(n \cdot T^{max}_{T_V(O)} \cdot T^{max}_{reached})$

Algorithm 2 Progression breadth-first search for a FDR planning task using FMs

function $\textsc{BfsProgFinal}(V, I, O, \gamma)$
  $T_V(O) \leftarrow \textsc{Formula}(\bigvee_{o \in O} \tau_V(o))$ ▷ Only needs to be computed once
  $goalStates \leftarrow \textsc{Formula}(\gamma)$ ▷ Only needs to be computed once
  $reached_0 \leftarrow \textsc{Singleton}(I)$ ▷ Only needs to be computed once
  $i \leftarrow 0$
  loop
    if $reached_i \cap goalStates \neq 0$ **then** ▷ Use Intersection, Equals and False
      return solution found
    $reached_{i+1} \leftarrow reached_i \cup \textsc{Apply}(reached_i, T_V(O))$ ▷ Use Union
    if $reached_{i+1} = reached_i$ **then** ▷ Use Equals
      return no solution exists
    $i \leftarrow i + 1$

Conclusion

- Operations inside loop run in polynomial time
- Usable search algorithm for FDR planning tasks using FMs

We can use FMs as knowledge compilation for symbolic search on FDR planning tasks.

Questions?

leonhard.badenberg@stud.unibas.ch

Factored Mappings

A Factored Mapping (FM) σ over V has

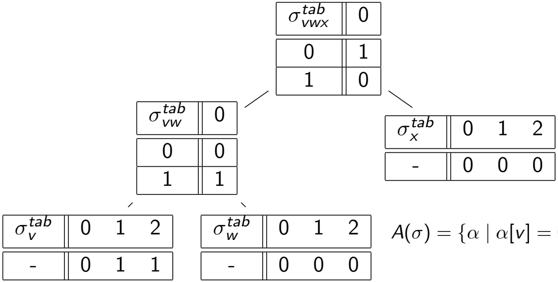
- a finite value set $vals(\sigma)$,
- an associated table σ^{tab}, and
- can be atomic or a merge.

Atomic: $\sigma^{tab} : dom(v) \rightarrow vals(\sigma)$

Merge: $\sigma^{tab} : vals(\sigma_L) \times vals(\sigma_R) \rightarrow vals(\sigma)$

Building Basic Factored Mappings

- False ($\perp$): $A(\sigma) = \emptyset$
  - Fills all entries in all tables with zeroes
- True ($\top$): $A(\sigma) = A(V)$
  - Fills all entries in all tables with zeroes
  - Sets entry in root table to 1



$A(\sigma) = \{\alpha \mid \alpha[v] = 0\}$

- Atom ($v = c$): $A(\sigma) = \{\alpha \mid \alpha[v] = c\}$
  - Fills σ^{tab}_v with 0 and 1, where $\sigma^{tab}_v(c) \neq \sigma^{tab}_v(d)$ for all $d \neq c$
  - Fills all other atomic tables with zeroes
  - Swaps root table entries if $\sigma^{tab}_v(c) = 0$

For all of these: $\mathcal{O}(n \cdot D^{max})$, where $n = |V|$ and $D^{max} = \max_{v \in V} |dom(v)|$