Merge-and-Shrink Task Reformulation for Classical Planning

Álvaro Torralba, Silvan Sievers

HSDIP 2019

Classical Planning

Definition. *A planning task is a 4-tuple* $\Pi = (V, A, I, G)$ *where:*

- V *is a set of state variables, each* $v \in V$ *with a finite domain* D_v*.*
- A *is a set of actions; each* $a \in A$ *is a triple* (pre_a, eff_a, c_a)*, of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}_0^+$*.*
- *Initial state* I *(complete assignment), goal* G *(partial assignment).*

$\rightarrow$ Solution ("Plan"): Action sequence mapping I into s s.t. $s \models G$.

Classical Planning

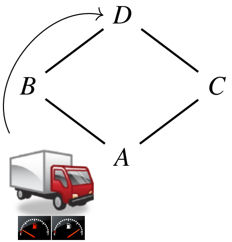
Definition. *A planning task is a 4-tuple $\Pi = (V, A, I, G)$ where:*

- V *is a set of state variables, each* $v \in V$ *with a finite domain* D_v.
- A *is a set of actions; each* $a \in A$ *is a triple* (pre_a, eff_a, c_a), *of precondition and effect (partial assignments), and the action's cost* $c_a \in \mathbb{R}_0^+$.
- *Initial state* I *(complete assignment), goal* G *(partial assignment).*

$\rightarrow$ Solution ("Plan"): Action sequence mapping I into s s.t. $s \models G$.

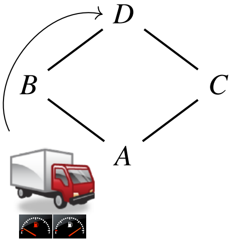
Running Example:

- $V = \{T, F\}$ with $D_t = \{A, B, C, D\}$, $D_F = \{0, 1, 2\}$.
- $A = \{drive(x, x', f, f')\}$
- $I = \{T = A, F = 2\}$
- $G = \{T = D\}$



Accidental Complexity

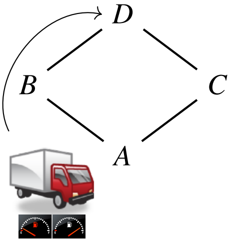
- $V = \{T, F\}$ with $D_t = \{A, B, C, D\}$, $D_F = \{0, 1, 2\}$.
- $A = \{drive(x, x', f, f')\}$
- $I = \{T = A, F = 2\}$
- $G = \{T = D\}$



Accidental complexity: when solving the problem is harder due to how it is encoded

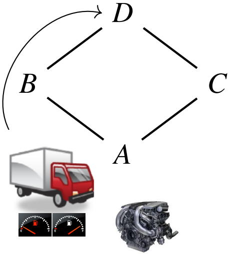
Accidental Complexity

- $V = \{T, F\}$ with $D_t = \{A, B, C, D\}$, $D_F = \{0, 1, 2\}$.
- $A = \{drive(x, x', f, f')\}$
- $I = \{T = A, F = 2\}$
- $G = \{T = D\}$



Accidental complexity: when solving the problem is harder due to how it is encoded

- $V = \{T, F, E\}$ with $D_t = \{A, B, C, D\}$, $D_F = \{0, 1, 2\}$, $D_E = \{on, off\}$.
- $A = \{drive(x, x', f, f'), turnon, turnoff\}$
- $I = \{T = A, F = 2, E = off\}$
- $G = \{T = D\}$



Reformulation

Transform the model to get rid of "accidental" complexity

Reformulation

Transform the model to get rid of “accidental” complexity

Task Π

Reformulation

Transform the model to get rid of "accidental" complexity

$$\text{Task } \Pi \xrightarrow{\text{reformulation } (\rho)} \text{Task } \rho(\Pi)$$

Reformulation

Transform the model to get rid of "accidental" complexity

$$
\begin{tikzcd}
\text{Task } \Pi \arrow[r, "\text{reformulation } (\rho)"] & \text{Task } \rho(\Pi) \arrow[d, "\text{solve}"] \\
& \text{Plan } \pi^{\rho}
\end{tikzcd}
$$

Reformulation

Transform the model to get rid of “accidental” complexity

$$
\begin{tikzcd}
\text{Task } \Pi \arrow[r, "\text{reformulation } (\rho)"] \arrow[d, dashed, no head] & \text{Task } \rho(\Pi) \arrow[d, "\text{solve}"] \\
\text{Plan } \pi = \overleftarrow{\rho}(\pi^\rho) & \text{Plan } \pi^\rho \arrow[l, "\text{plan reconstruction } (\overleftarrow{\rho})"']
\end{tikzcd}
$$

Reformulation

Transform the model to get rid of “accidental” complexity

$$\begin{tikzcd}
\text{Task } \Pi \arrow[rr, "\text{reformulation } (\rho)"] \arrow[d, dash, dashed] & & \text{Task } \rho(\Pi) \arrow[d, "\text{solve}"] \\
\text{Plan } \pi = \overleftarrow{\rho}(\pi^\rho) & & \text{Plan } \pi^\rho \arrow[ll, "\text{plan reconstruction } (\overleftarrow{\rho})"']
\end{tikzcd}$$

Properties:

- Polynomial: ρ and $\overleftarrow{\rho}$ run in polynomial time in the $|\Pi|$ and $|\overleftarrow{\rho}(\pi^\rho)|$.

Reformulation

Transform the model to get rid of "accidental" complexity

$$\begin{tikzcd}
\text{Task } \Pi \arrow[r, "\text{reformulation } (\rho)"] \arrow[d, dash, dashed] & \text{Task } \rho(\Pi) \arrow[d, "\text{solve}"] \\
\text{Plan } \pi = \overleftarrow{\rho}(\pi^\rho) & \text{Plan } \pi^\rho \arrow[l, "\text{plan reconstruction } (\overleftarrow{\rho})"']
\end{tikzcd}$$

Properties:

- Polynomial: ρ and $\overleftarrow{\rho}$ run in polynomial time in the $|\Pi|$ and $|\overleftarrow{\rho}(\pi^\rho)|$.
- Optimal: π^ρ is optimal for $\rho(\Pi) \Rightarrow \overleftarrow{\rho}(\pi^\rho)$

Abstraction Heuristics

Θ

Abstraction Heuristics

Θ

Abstraction Heuristics

$$\Theta \xrightarrow{\text{abstraction } \alpha} \Theta^{\alpha}$$

Abstraction Heuristics

$$
\begin{tikzcd}
\Theta \arrow[rr, "\text{abstraction } \alpha"] & & \Theta^\alpha \arrow[d, "\text{solve}"] \\
& & h^\alpha
\end{tikzcd}
$$

Abstraction Heuristics

$$
\begin{tikzcd}
\Theta \arrow[rr, "\text{abstraction } \alpha"] \arrow[d, dash, dashed] & & \Theta^{\alpha} \arrow[d, "\text{solve}"] \\
h & & h^{\alpha} \arrow[ll, "\text{estimates}"']
\end{tikzcd}
$$

Abstraction Heuristics

$$\begin{tikzcd}
\Theta \arrow[r, "\text{abstraction } \alpha"] \arrow[d, dash, dashed] & \Theta^\alpha \arrow[d, "\text{solve}"] \\
h & h^\alpha \arrow[l, "\text{estimates}"']
\end{tikzcd}$$

$\rightarrow$An abstraction is refinable if a solution for the abstract task can be transformed in polynomial time in a solution for the original

FDR Reformulation

Free DTG: The domain transition graph of a variable only considering actions without preconditions or effects on other variables.

Variable Abstraction (Helmert, 2006)

Any variable whose free DTG is strongly connected can be abstracted away

FDR Reformulation

Free DTG: The domain transition graph of a variable only considering actions without preconditions or effects on other variables.

Variable Abstraction (Helmert, 2006)

Any variable whose free DTG is strongly connected can be abstracted away

→In our example: gets rid of variable E

$$\begin{aligned} \text{Abstract plan} &= \langle \mathrm{DR}_{\text{A-B},2\text{-}1}, \mathrm{DR}_{\text{B-D},1\text{-}0} \rangle \\ \text{Original plan} &= \langle \end{aligned}$$

FDR Reformulation

Free DTG: The domain transition graph of a variable only considering actions without preconditions or effects on other variables.

Variable Abstraction (Helmert, 2006)

Any variable whose free DTG is strongly connected can be abstracted away

$\rightarrow$In our example: gets rid of variable E

$$\begin{aligned} \text{Abstract plan} &= \langle \mathrm{DR}_{\text{A-B},2\text{-}1}, \mathrm{DR}_{\text{B-D},1\text{-}0} \rangle \\ \text{Original plan} &= \langle \mathit{turnon}, \mathrm{DR}_{\text{A-B},2\text{-}1}, \mathrm{DR}_{\text{B-D},1\text{-}0} \rangle \end{aligned}$$

FDR Reformulation

Free DTG: The domain transition graph of a variable only considering actions without preconditions or effects on other variables.

Variable Abstraction (Helmert, 2006)

Any variable whose free DTG is strongly connected can be abstracted away

$\rightarrow$In our example: gets rid of variable E

$$\begin{aligned} \text{Abstract plan} &= \langle \text{DR}_{\text{A-B},2\text{-}1}, \text{DR}_{\text{B-D},1\text{-}0} \rangle \\ \text{Original plan} &= \langle \mathit{turnon}, \text{DR}_{\text{A-B},2\text{-}1}, \text{DR}_{\text{B-D},1\text{-}0} \rangle \end{aligned}$$

$\rightarrow$In Logistics: solves the problem without doing any search

FDR Reformulation

Free DTG: The domain transition graph of a variable only considering actions without preconditions or effects on other variables.

Variable Abstraction (Helmert, 2006)

Any variable whose free DTG is strongly connected can be abstracted away

$\rightarrow$In our example: gets rid of variable E

$$\begin{aligned} \text{Abstract plan} &= \langle \mathrm{DR}_{\text{A-B},2\text{-}1}, \mathrm{DR}_{\text{B-D},1\text{-}0} \rangle \\ \text{Original plan} &= \langle \mathit{turnon}, \mathrm{DR}_{\text{A-B},2\text{-}1}, \mathrm{DR}_{\text{B-D},1\text{-}0} \rangle \end{aligned}$$

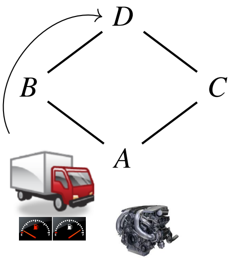
$\rightarrow$In Logistics: solves the problem without doing any search

Extensions:

- Haslum (2007) gave a stronger criteria
- Tozicka et al. (2016) use this to merge values of a variable

Running Example with Accidental Complexity

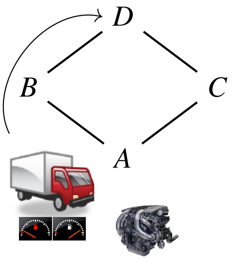
- $V = \{T, F, E\}$ with $D_t = \{A, B, C, D\}$, $D_F = \{0, 1, 2\}$, $D_E = \{\mathit{off}, \mathit{rd}, \mathit{on}\}$.
- $A = \{\mathit{drive}(x, x', f, f'), \mathit{turnon}, \mathit{checkfuel}\}$
- $I = \{T = A, F = 2, E = \mathit{off}\}$
- $G = \{T = D\}$



$\rightarrow$Before turning on the engine, we need to check that we have 2 units of fuel with the check-fuel action

Running Example with Accidental Complexity

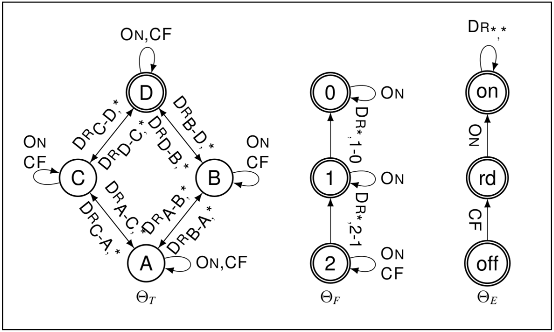
- $V = \{T, F, E\}$ with $D_t = \{A, B, C, D\}$, $D_F = \{0, 1, 2\}$, $D_E = \{\mathit{off}, \mathit{rd}, \mathit{on}\}$.
- $A = \{\mathit{drive}(x, x', f, f'), \mathit{turnon}, \mathit{checkfuel}\}$
- $I = \{T = A, F = 2, E = \mathit{off}\}$
- $G = \{T = D\}$



$\rightarrow$Before turning on the engine, we need to check that we have 2 units of fuel with the check-fuel action

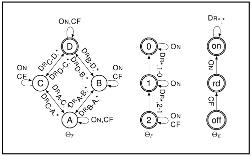
- check-fuel:
  pre: E=off, F=2
  eff: E=rd
- turn-on:
  pre: E=rd
  eff: E=on

Merge-and-Shrink



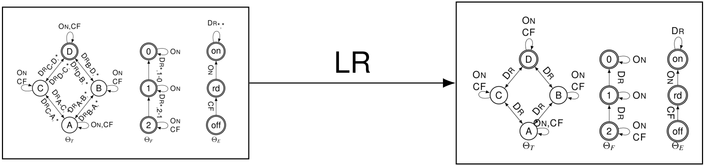
Atomic: One TS per variable such that $\Theta_1 \otimes \Theta_2 \otimes \Theta_3 = \Theta$

Merge-and-Shrink



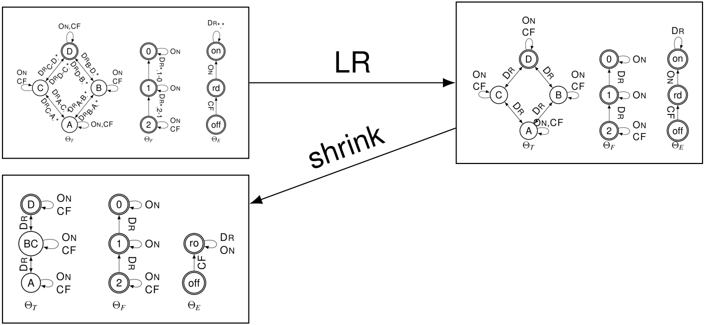
Atomic: One TS per variable such that $\Theta_1 \otimes \Theta_2 \otimes \Theta_3 = \Theta$

Merge-and-Shrink



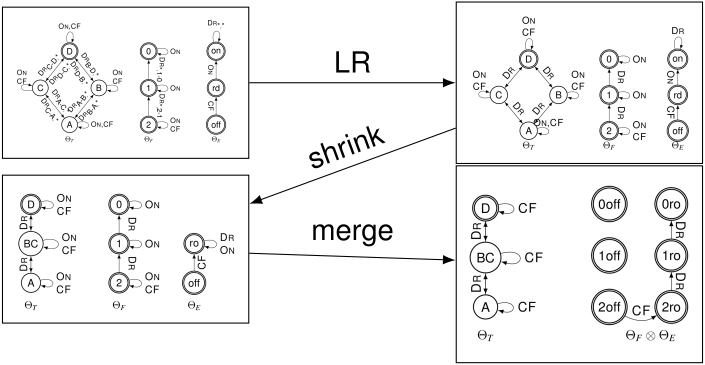
Atomic: One TS per variable such that $\Theta_1 \otimes \Theta_2 \otimes \Theta_3 = \Theta$

Merge-and-Shrink



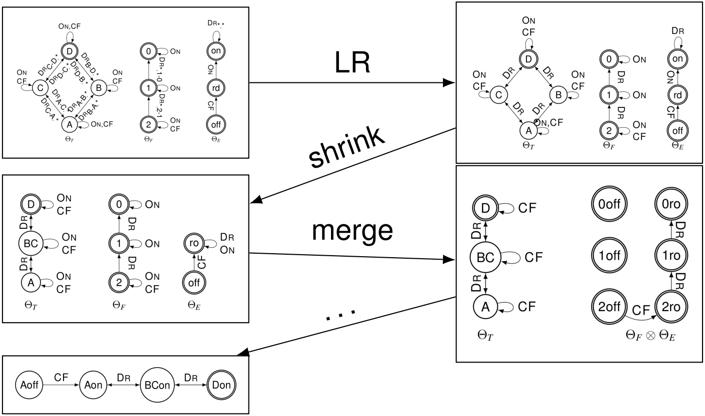
Atomic: One TS per variable such that $\Theta_1 \otimes \Theta_2 \otimes \Theta_3 = \Theta$

Merge-and-Shrink



Atomic: One TS per variable such that $\Theta_1 \otimes \Theta_2 \otimes \Theta_3 = \Theta$

Merge-and-Shrink



Atomic: One TS per variable such that $\Theta_1 \otimes \Theta_2 \otimes \Theta_3 = \Theta$

M&S as Reformulation Framework

$$\begin{tikzcd}
\Pi^{FDR} \arrow[r] \arrow[drr, bend right] & \{\Theta_1, \dots, \Theta_n\}, L \arrow[r, "\text{M\&S}"] & \{\Theta^\alpha\} \arrow[d, dashed, "\text{h}"] \\
 & & \text{Search}
\end{tikzcd}$$

M&S as Reformulation Framework

$$
\begin{tikzcd}
\Pi^{FDR} \arrow[r] & \{\Theta_1, \dots, \Theta_n\}, L \arrow[d, "\text{M\&S}"'] & \{\Theta^\alpha\} \arrow[d, dashed, "\text{h}"] \\
& \{\Theta'_1, \dots, \Theta'_k\}, L' \arrow[r] \arrow[ur, "\text{M\&S}"] & \text{Search}
\end{tikzcd}
$$

M&S as Reformulation Framework

$$\begin{tikzcd}
\Pi^{FDR} \arrow[r] & \{\Theta_1, \ldots, \Theta_n\}, L \arrow[d, "\text{M\&S}"'] \arrow[r, phantom] & \{\Theta^\alpha\} \arrow[d, dashed, "\text{h}"] \\
& \{\Theta'_1, \ldots, \Theta'_k\}, L' \arrow[ur, "\text{M\&S}"] \arrow[r] & \text{Search}
\end{tikzcd}$$

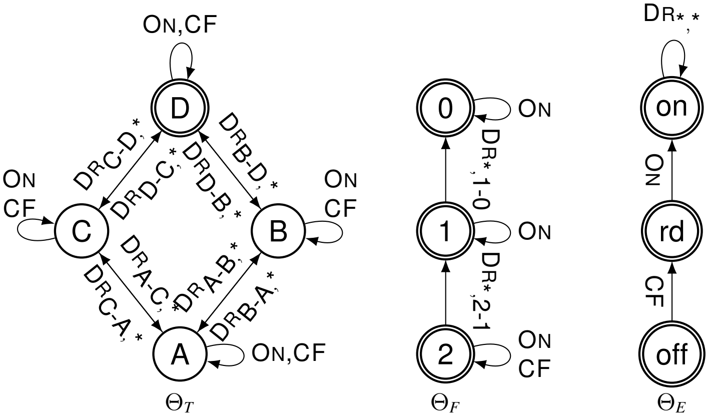
$$\begin{tikzcd}
\Pi^{FDR} \arrow[rr, "\text{delete relaxation}"] \arrow[drr, bend right] & & h^{FF} \arrow[d, dashed, "\text{h}"] \\
& & \text{Search}
\end{tikzcd}$$

M&S as Reformulation Framework

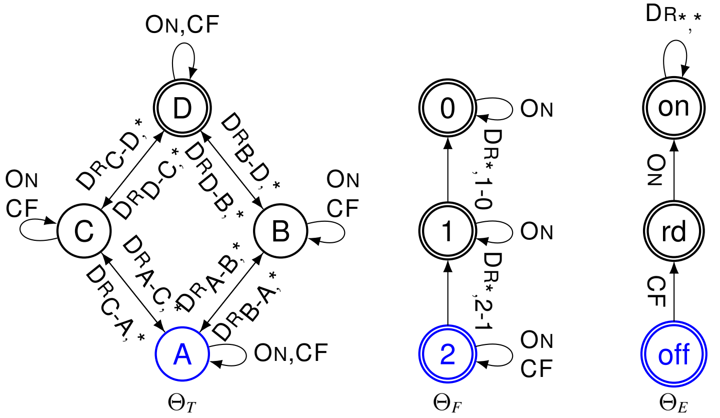
$$
\begin{tikzcd}
\Pi^{FDR} \arrow[r] & \{\Theta_1, \dots, \Theta_n\}, L \arrow[d, "\text{M\&S}"'] & \{\Theta^{\alpha}\} \arrow[d, dashed, "\text{h}"] \\
& \{\Theta'_1, \dots, \Theta'_k\}, L' \arrow[ur, "\text{M\&S}"] \arrow[r] & \text{Search}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\Pi^{FDR} \arrow[r] & \{\Theta_1, \dots, \Theta_n\}, L \arrow[d, "\text{M\&S}"'] & h^{FF} \arrow[d, dashed, "\text{h}"] \\
& \{\Theta'_1, \dots, \Theta'_k\}, L' \arrow[ur, "\text{delete relaxation}"] \arrow[r] & \text{Search}
\end{tikzcd}
$$

FTS Representation and Successor Generation

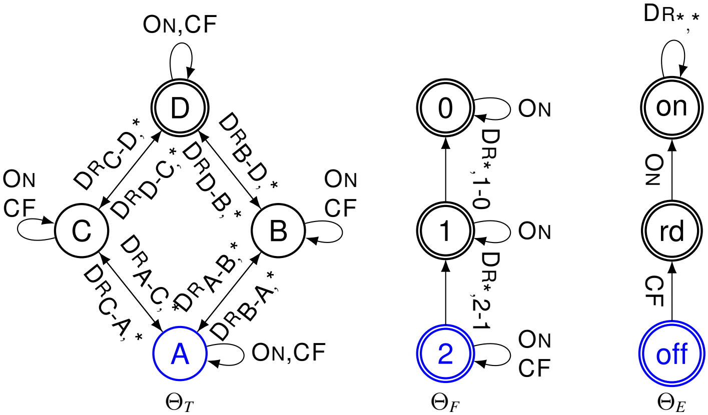


FTS Representation and Successor Generation



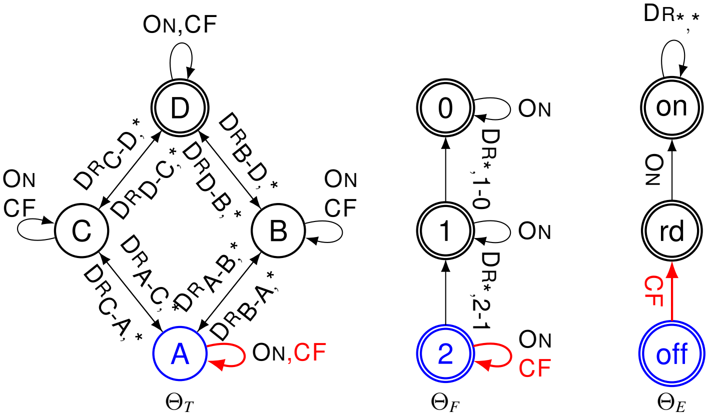
State: ⟨T=A, F=2, E=off⟩

FTS Representation and Successor Generation



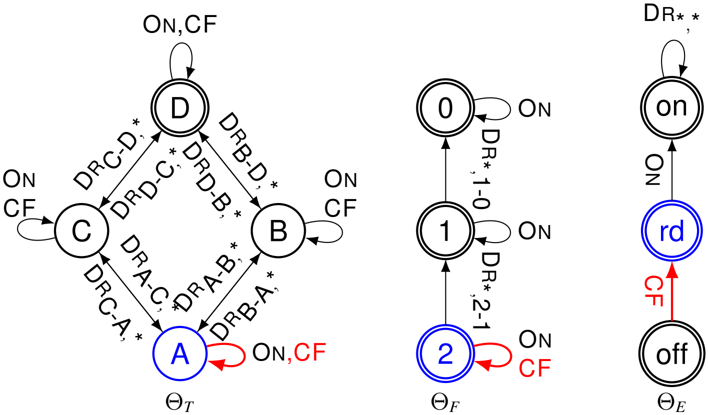
State: ⟨T=A, F=2, E=off⟩
Applicable labels:

FTS Representation and Successor Generation



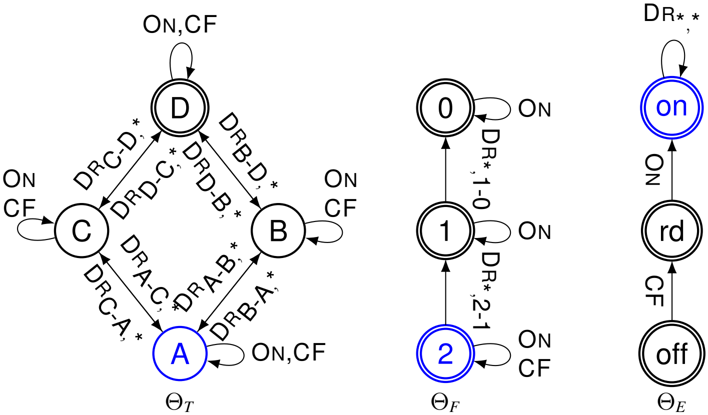
State: ⟨T=A, F=2, E=off⟩
Applicable labels: $\{CF\}$

FTS Representation and Successor Generation



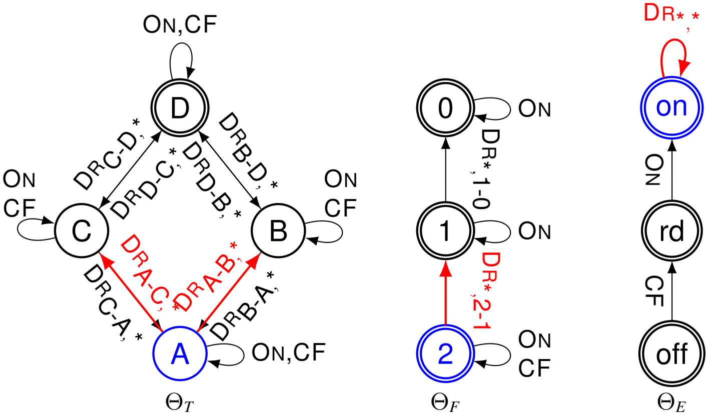
State: ⟨T=A, F=2, E=off⟩
Applicable labels: $\{CF\}$
Successor: ⟨T=A, F=2, E=rd⟩

FTS Representation and Successor Generation



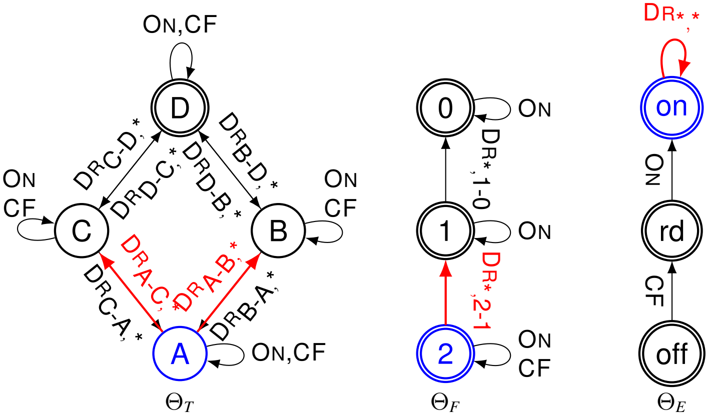
State: ⟨T=A, F=2, E=on⟩

FTS Representation and Successor Generation



State: $\langle$T=A, F=2, E=on$\rangle$
Applicable labels: $\{\text{DR}_{\text{A-B,2-1}}, \text{DR}_{\text{A-C,2-1}}\}$

FTS Representation and Successor Generation



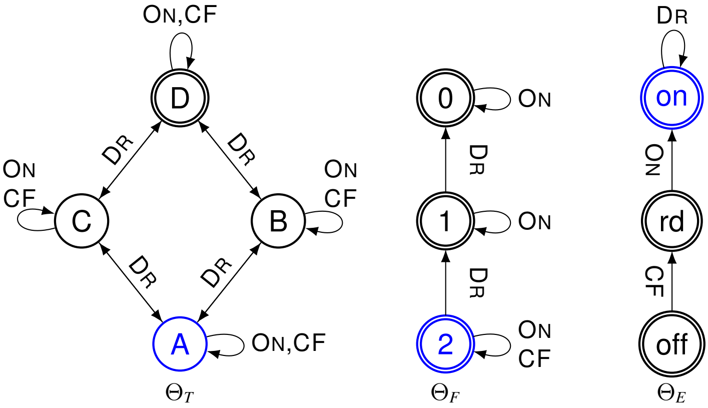
State: ⟨T=A, F=2, E=on⟩

Applicable labels: $\{\text{DR}_{\text{A-B},2\text{-}1}, \text{DR}_{\text{A-C},2\text{-}1}\}$

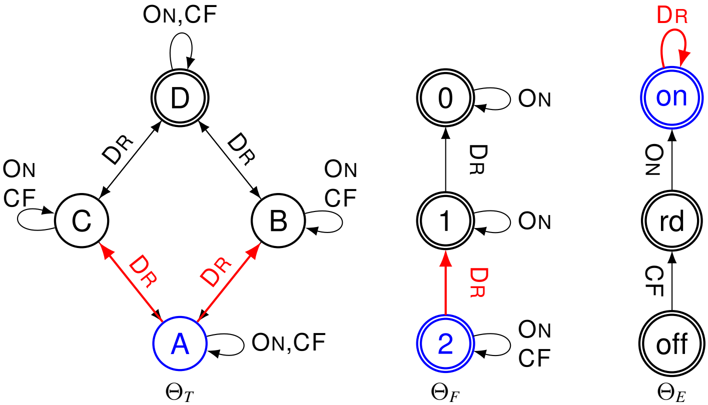
Successor ($\text{DR}_{\text{A-B},2\text{-}1}$): ⟨T=B, F=1, E=on⟩

Successor ($\text{DR}_{\text{A-C},2\text{-}1}$): ⟨T=C, F=1, E=on⟩

FTS Representation and Successor Generation



FTS Representation and Successor Generation



State: ⟨T=A, F=2, E=on⟩
Applicable labels: {DR}

Successor (DR): ⟨T=B, F=1, E=on⟩

Successor (DR): ⟨T=C, F=1, E=on⟩

What's the Difference Anyway?

Advantage of FTS over FDR:

- Limited form of disjunctive preconditions
- Limited form of conditional effects
- Limited form of non-deterministic effects

What's the Difference Anyway?

Advantage of FTS over FDR:

- Limited form of disjunctive preconditions
- Limited form of conditional effects
- Limited form of non-deterministic effects

[1]2blue1 [1]2green1 [1]2lightgray1 [1]2dkgreen1 [1]2orange1 [

What's the Difference Anyway?

Advantage of FTS over FDR:

- Limited form of disjunctive preconditions
- Limited form of conditional effects
- Limited form of non-deterministic effects

[1]2blue1 [1]2green1 [1]2lightgray1 [1]2dkgreen1 [1]2orange1 [

What's the Difference Anyway?

Advantage of FTS over FDR:

- Limited form of disjunctive preconditions
- Limited form of conditional effects
- Limited form of non-deterministic effects

[1]2blue1 [1]2green1 [1]2lightgray1 [1]2dkgreen1 [1]2orange1 [

What's the Difference Anyway?

Advantage of FTS over FDR:

- Limited form of disjunctive preconditions
- Limited form of conditional effects
- Limited form of non-deterministic effects

[1]2blue1 [1]2green1 [1]2lightgray1 [1]2dkgreen1 [1]2orange1 [

What's the Difference Anyway?

Advantage of FTS over FDR:

- Limited form of disjunctive preconditions
- Limited form of conditional effects
- Limited form of non-deterministic effects

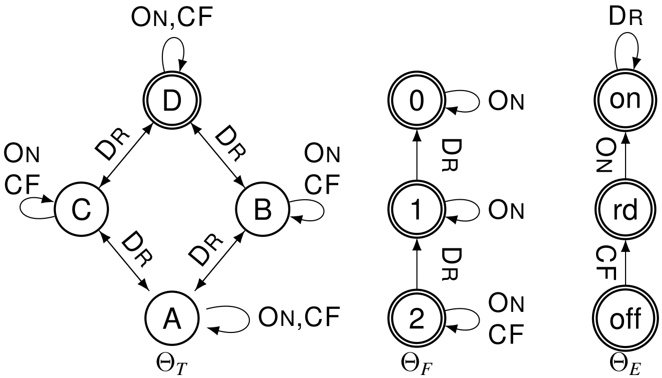
[1]2blue1 [1]2green1 [1]2lightgray1 [1]2dkgreen1 [1]2orange1 [

Shrink

Replaces a TS Θ_i by an abstraction thereof ($\alpha(\Theta_i)$)
The state space of the new task is an abstraction of the original

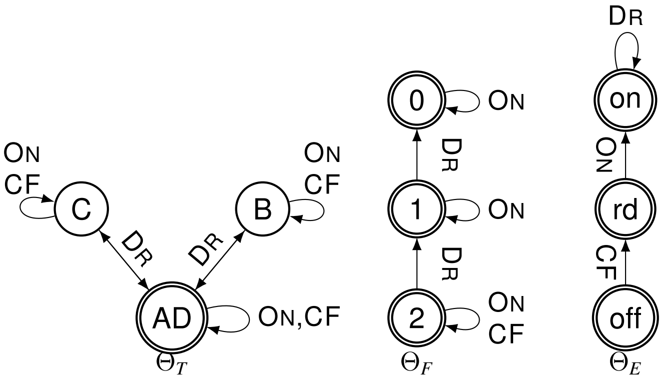
Shrink

Replaces a TS Θ_i by an abstraction thereof ($\alpha(\Theta_i)$)
The state space of the new task is an abstraction of the original



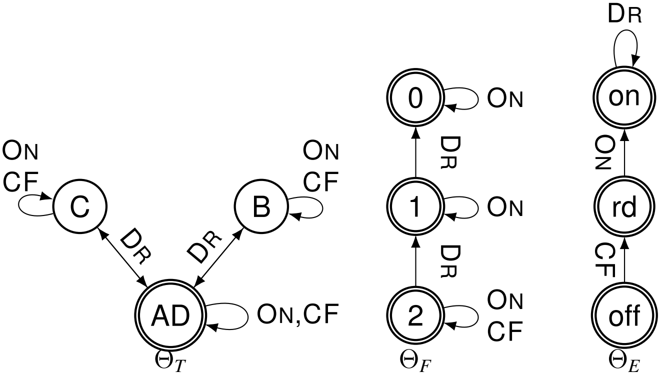
Shrink

Replaces a TS Θ_i by an abstraction thereof ($\alpha(\Theta_i)$)
The state space of the new task is an abstraction of the original



Shrink

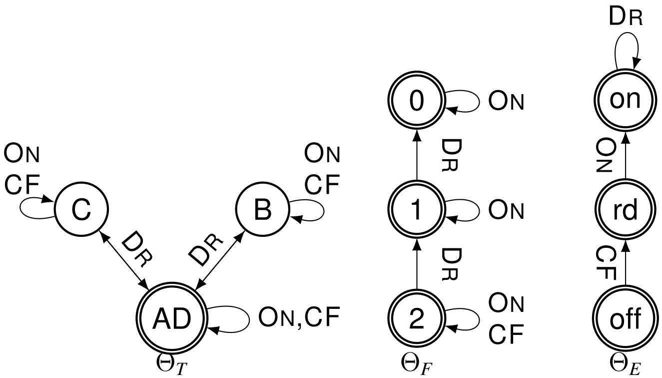
Replaces a TS Θ_i by an abstraction thereof ($\alpha(\Theta_i)$)
The state space of the new task is an abstraction of the original



Plan = $\langle\rangle$

Shrink

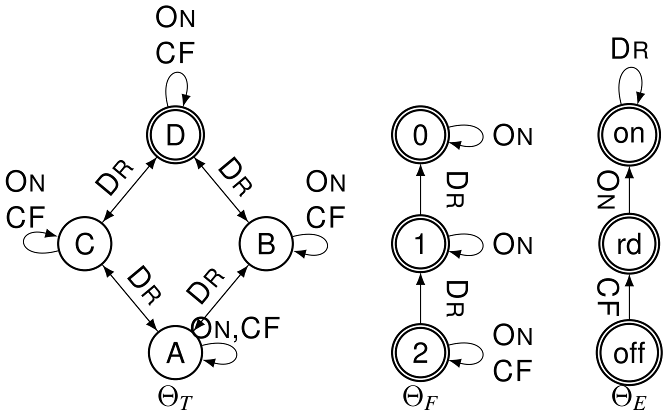
Replaces a TS Θ_i by an abstraction thereof ($\alpha(\Theta_i)$)
The state space of the new task is an abstraction of the original



Plan = $\langle\rangle \Rightarrow$ Only "refinable" abstractions are suitable for reformulation

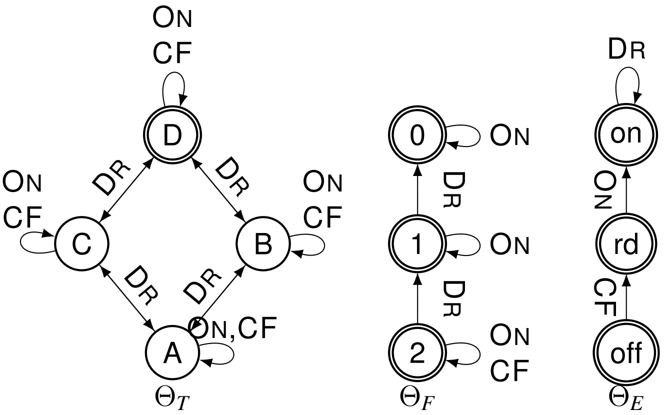
(Strong) Bisimulation Shrinking

Two states are equivalent if they have the same outgoing transitions



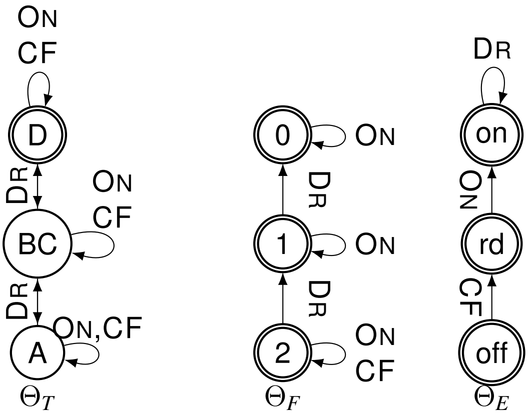
(Strong) Bisimulation Shrinking

Two states are equivalent if they have the same outgoing transitions



(Strong) Bisimulation Shrinking

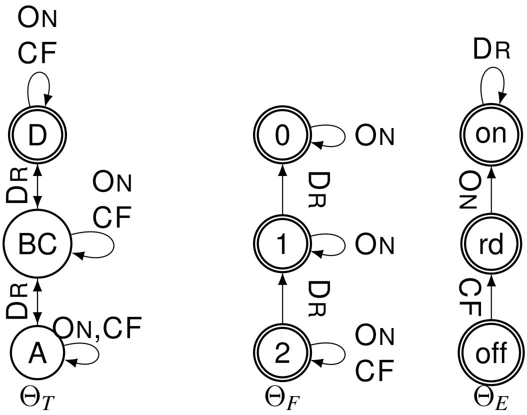
Two states are equivalent if they have the same outgoing transitions



Plan = $\langle$CF, ON, DR(BC), DR(D)$\rangle \rightarrow$

(Strong) Bisimulation Shrinking

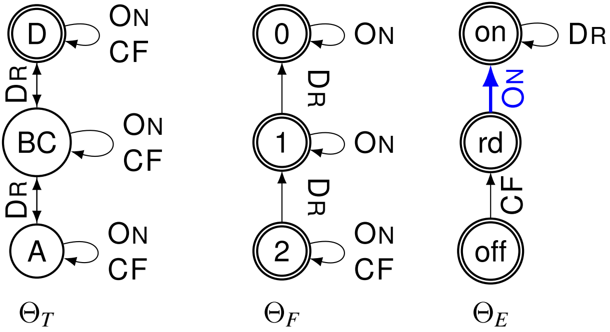
Two states are equivalent if they have the same outgoing transitions



Plan = $\langle$CF, ON, DR(BC), DR$(D)\rangle \rightarrow \langle$CF, ON, DR$(B)$, DR$(D)\rangle$

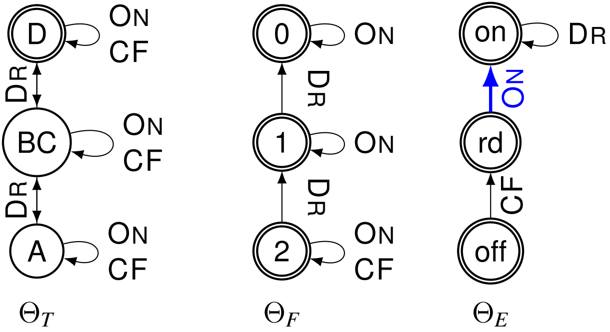
Weak Bisimulation Shrinking

1. Identify τ labels that are internal to a TS (self-loop everywhere)



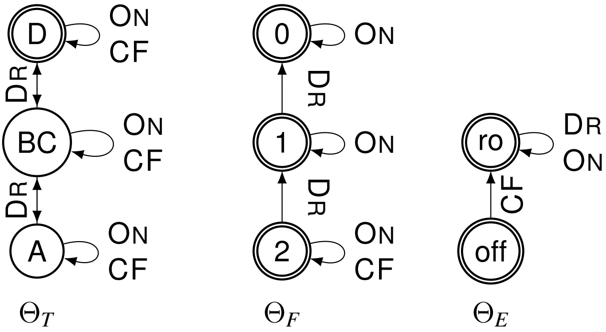
Weak Bisimulation Shrinking

1. Identify τ labels that are internal to a TS (self-loop everywhere)
2. Bisimulation allowing free-use of τ labels



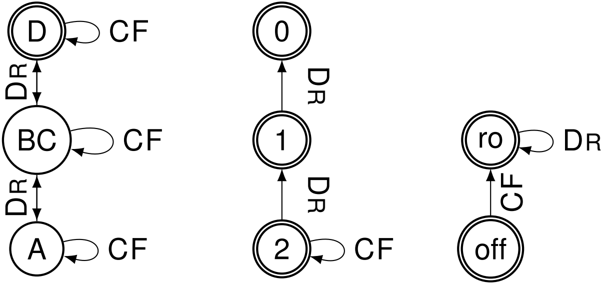
Weak Bisimulation Shrinking

1. Identify τ labels that are internal to a TS (self-loop everywhere)
2. Bisimulation allowing free-use of τ labels



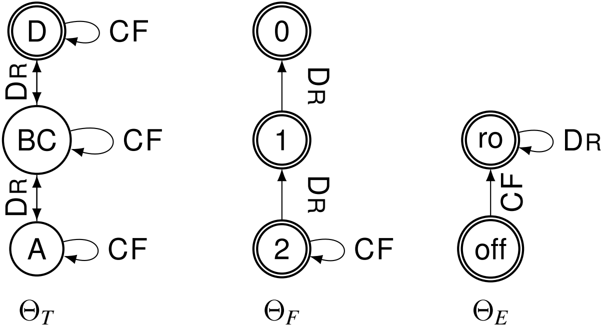
Weak Bisimulation Shrinking

1. Identify τ labels that are internal to a TS (self-loop everywhere)
2. Bisimulation allowing free-use of τ labels



Weak Bisimulation Shrinking

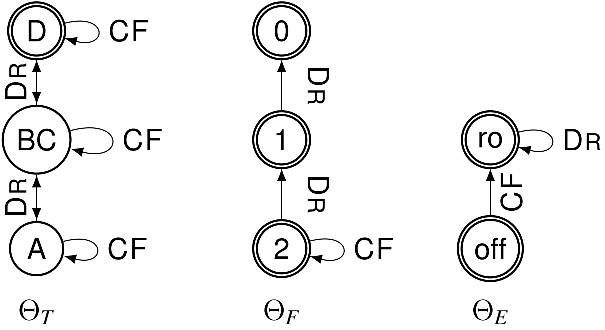
1. Identify τ labels that are internal to a TS (self-loop everywhere)
2. Bisimulation allowing free-use of τ labels



Plan = $\langle$CF, DR(BC), DR$(D)\rangle \rightarrow$

Weak Bisimulation Shrinking

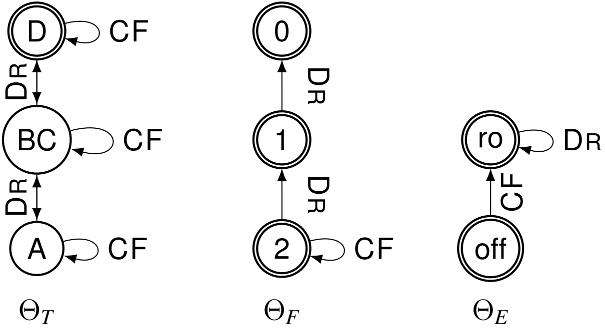
1. Identify τ labels that are internal to a TS (self-loop everywhere)
2. Bisimulation allowing free-use of τ labels



Plan = $\langle$CF, DR(BC), DR$(D)\rangle \rightarrow \langle$CF, ON, DR$(BC)$, DR$(D)\rangle$

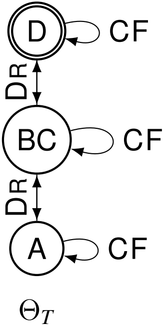
Merge

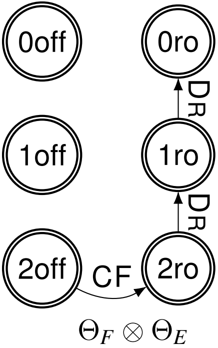
Replace Θ_i and Θ_j by their product: $\Theta_i \otimes \Theta_j$



Merge

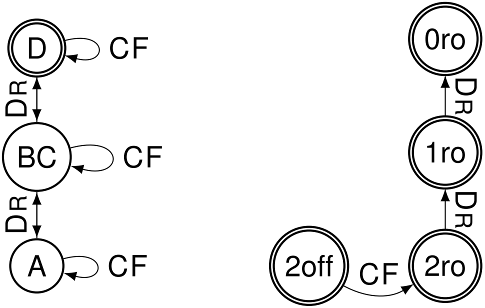
Replace Θ_i and Θ_j by their product: $\Theta_i \otimes \Theta_j$





Merge

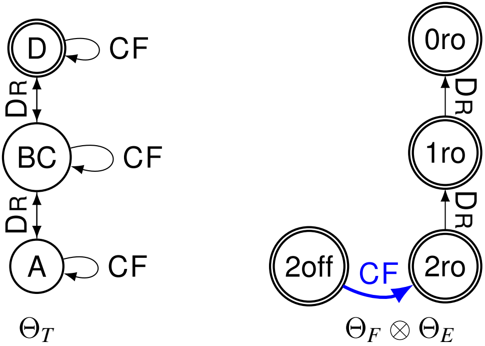
Replace Θ_i and Θ_j by their product: $\Theta_i \otimes \Theta_j$



Θ_T $\Theta_F \otimes \Theta_E$

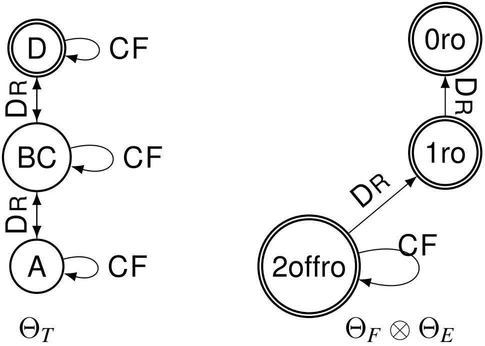
Merge

Replace Θ_i and Θ_j by their product: $\Theta_i \otimes \Theta_j$



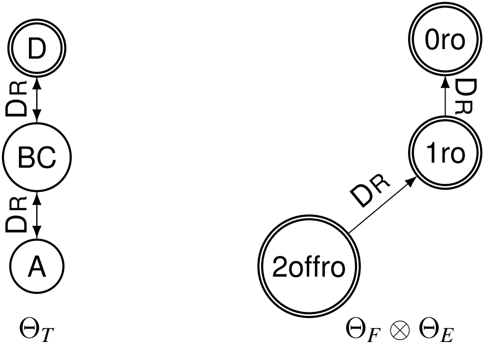
Merge

Replace Θ_i and Θ_j by their product: $\Theta_i \otimes \Theta_j$



Merge

Replace Θ_i and Θ_j by their product: $\Theta_i \otimes \Theta_j$



Relation to FDR Reformulation Methods

An FTS reformulation method dominates an FDR reformulation method if it can do the same reformulations:

$$\begin{tikzcd}
\Pi \arrow[rr, "\rho^{FDR}"] \arrow[d, "\text{atomic}"] & & \rho^{FDR}(\Pi) \arrow[d, "\text{atomic}"] \\
\{\Theta_1, \ldots, \Theta_n\}, L & & \{\Theta'_1, \ldots, \Theta'_m\}, L'
\end{tikzcd}$$

Relation to FDR Reformulation Methods

An FTS reformulation method dominates an FDR reformulation method if it can do the same reformulations:

$$\begin{tikzcd}
\Pi \arrow[rr, "\rho^{FDR}"] \arrow[d, "\text{atomic}"] & & \rho^{FDR}(\Pi) \arrow[d, "\text{atomic}"] \\
\{\Theta_1, \dots, \Theta_n\}, L \arrow[rr, "\rho^{FTS}"] & & \{\Theta'_1, \dots, \Theta'_m\}, L'
\end{tikzcd}$$

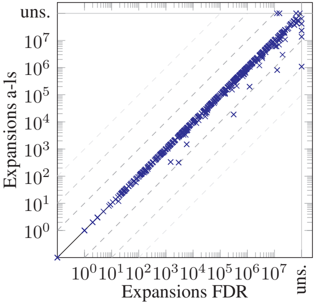
Relation to FDR Reformulation Methods

An FTS reformulation method **dominates** an FDR reformulation method if it **can do the same reformulations**:

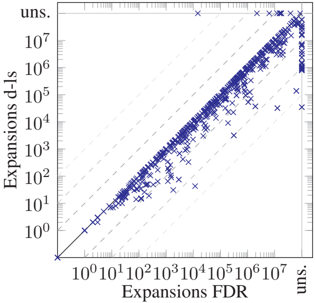
$$\begin{tikzcd}
\Pi \arrow[r, "\rho^{FDR}"] \arrow[d, "\text{atomic}"] & \rho^{FDR}(\Pi) \arrow[d, "\text{atomic}"] \\
\{\Theta_1, \ldots, \Theta_n\}, L \arrow[r, "\rho^{FTS}"] & \{\Theta'_1, \ldots, \Theta'_m\}, L'
\end{tikzcd}$$

- Variable abstraction and merge values are dominated by weak bisimulation shrinking (plus removing TSs with a core state)
- Generalize actions is dominated by label reduction

Search Space Reduction: Optimal

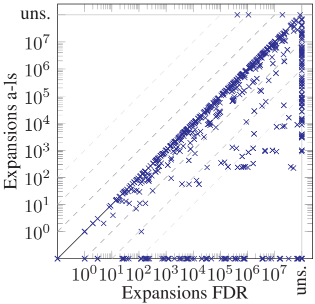


Bisimulation + LR

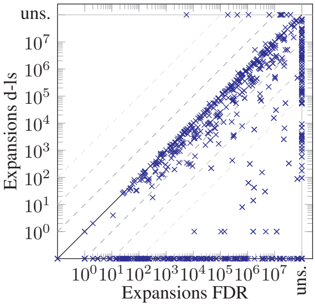


Merge (DFP) + LR + Bisimulation

Search Space Reduction: Satisficing



Weak Bisimulation + LR



Merge (DFP) + LR + Weak Bisimulation

Optimal Planning

	FDR	a	a-ls	d-ls	m-ls	tot	orcl
FDR						797	$h^{\max}$: 801
a						770	
a-ls						780	
d-ls					600		
m-ls					632		
FDR						822	$h^{\text{M\&S}_d}$: 910
a						826	
a-ls						831	
d-ls					815		
m-ls					849		

Optimal Planning

	FDR	a	a-ls	d-ls	m-ls	tot	orcl
FDR						797	$h^{\max}$: 801
a						770	
a-ls		**4**				780	
d-ls		2	1			600	
m-ls		4	4			632	
FDR						822	$h^{\mathrm{M\&S_d}}$: 910
a						826	
a-ls		**4**				831	
d-ls		11	10			815	
m-ls		15	15			849	

Optimal Planning

	FDR	a	a-ls	d-ls	m-ls	tot	orcl
FDR	–	**12**	**13**	**37**	**36**	797	$h^{\max}$: 801
a	1	–	1	**36**	**36**	770	
a-ls	3	**4**	–	**36**	**35**	780	
d-ls	2	2	1	–	7	600	
m-ls	4	4	4	**19**	–	632	
FDR	–	2	3	12	14	822	$h^{\mathrm{M\&S}_d}$: 910
a	**4**	–	1	**13**	**16**	826	
a-ls	**7**	**4**	–	**13**	**16**	831	
d-ls	**13**	11	10	–	11	815	
m-ls	**16**	15	15	**16**	–	849	

Satisficing Planning

	FDR	a	a-ls	d-ls	m-ls	tot	orcl
FDR						1326	h^{FF}: 1413
a						1272	
a-ls						1368	
d-ls					1208		
m-ls					1224		
FDR						1502	$h^{\mathrm{FF}}_{\mathrm{p.}}$: 1589
a						1461	
a-ls						1471	
d-ls					1357		
m-ls					1322		

Satisficing Planning

	FDR	a	a-ls	d-ls	m-ls	tot	orcl
FDR						1326	h^{FF}: 1413
a						1272	
a-ls		**15**				1368	
d-ls		10	4			1208	
m-ls		15	7			1224	
FDR						1502	h^{FF}_{p}: 1589
a						1461	
a-ls		8				1471	
d-ls		6	2			1357	
m-ls		7	3			1322	

Satisficing Planning

	FDR	a	a-ls	d-ls	m-ls	tot	orcl
FDR	–	**18**	15	**27**	**22**	1326	h^{FF}: 1413
a	6	–	13	**28**	**22**	1272	
a-ls	**18**	**15**	–	**31**	**24**	1368	
d-ls	10	10	4	–	11	1208	
m-ls	13	15	7	**21**	–	1224	
FDR	–	**17**	**15**	**24**	**23**	1502	$h^{FF}_{p.}$: 1589
a	8	–	**11**	**25**	**24**	1461	
a-ls	13	8	–	**26**	**26**	1471	
d-ls	9	6	2	–	15	1357	
m-ls	9	7	3	**16**	–	1322	

Conclusion

- Task reformulation is an important tool to solve planning tasks
- Merge-and-Shrink is a powerful reformulation framework
  - $\rightarrow$ dominates similar methods in FDR

Conclusion

- Task reformulation is an important tool to solve planning tasks
- Merge-and-Shrink is a powerful reformulation framework
  - $\rightarrow$ dominates similar methods in FDR
- Adapt search algorithms and heuristics for FTS
  - Successor Generation
  - Delete-relaxation heuristics (h^{FF})
  - $\rightarrow$ More abstraction heuristics for cost-optimal planning
  - $\rightarrow$ Landmarks and Novelty for satisficing planning

Conclusion

- Task reformulation is an important tool to solve planning tasks
- Merge-and-Shrink is a powerful reformulation framework
  - $\rightarrow$ dominates similar methods in FDR
- Adapt search algorithms and heuristics for FTS
  - Successor Generation
  - Delete-relaxation heuristics (h^{FF})
  - $\rightarrow$ More abstraction heuristics for cost-optimal planning
  - $\rightarrow$ Landmarks and Novelty for satisficing planning
- Provide a plan reconstruction for M&S transformations
  - Merge, LR, Pruning, Bisimulation $\rightarrow$ optimal reformulation
  - Weak bisimulation $\rightarrow$ satisficing reformulation
  - $\rightarrow$ Dominance-based pruning
  - $\rightarrow$ Tunnel macros