Иерархическое планирование и графовые грамматики

Тихон Пшеницын
МИАН

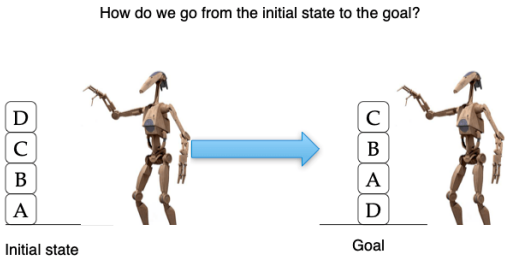
Однодневный семинар по математической логике
29 июня 2026 года

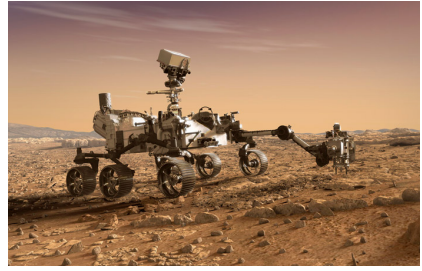
Это обзорный доклад!
Будет не только об иерархическом планировании,
но и о планировании в целом.

Автоматическое планирование

Неформальная постановка задачи

Агент хочет достичь поставленной цели. Для этого он может выполнять действия из определенного набора. Задача — составить план действий.





Классическое планирование (STRIPS)

Дана конечная первопорядковая сигнатура Ω. $\mathrm{At}(\Omega)$ — множество всех атомарных формул над Ω. Состояние — множество замкнутых атомарных формул.

Определение

Задача планирования STRIPS (‘**ST**anford **R**esearch **I**nstitute **P**roblem **S**olver’, 1971) — набор $\langle \mathcal{OP}, pre_+, pre_-, del, add, Init, Goal \rangle$, где

$\mathcal{OP}$ — конечное множество операций,

$pre_+, pre_-, del, add : \mathcal{OP} \to 2^{\mathrm{At}(\Omega)}$,

$Init, Goal \in 2^{\mathrm{At}(\Omega)}$ — начальное и заключительное состояния.

Пример

Операция `grab`:

- $pre_+(\texttt{grab}) = \{\texttt{on\_ground}(x)\}$
- $pre_-(\texttt{grab}) = \{\texttt{too\_far}(x), \texttt{full}(\texttt{hand})\}$
- $del(\texttt{grab}) = \{\texttt{on\_ground}(x)\}$
- $add(\texttt{grab}) = \{\texttt{in\_hand}(x), \texttt{full}(\texttt{hand})\}$

Классическое планирование (STRIPS)

Дана конечная первопорядковая сигнатура Ω. $\mathrm{At}(\Omega)$ — множество всех атомарных формул над Ω. Состояние — множество замкнутых атомарных формул.

Определение

Задача планирования STRIPS (‘**ST**anford **R**esearch **I**nstitute **P**roblem **S**olver’, 1971) — набор $\langle \mathcal{OP}, pre_+, pre_-, del, add, Init, Goal \rangle$, где

$\mathcal{OP}$ — конечное множество операций,

$pre_+, pre_-, del, add : \mathcal{OP} \to 2^{\mathrm{At}(\Omega)}$,

$Init, Goal \in 2^{\mathrm{At}(\Omega)}$ — начальное и заключительное состояния.

Определение

Выполнение операции $a \in \mathcal{OP}$ в состоянии s возможно, если существует подстановка θ, такая что $pre_+(a)\theta \subseteq s$ и $pre_-(a)\theta \cap s = \emptyset$. Результат выполнения — состояние $(s \setminus del(a)\theta) \cup add(a)\theta$.

Классическое планирование (STRIPS)

Дана конечная первопорядковая сигнатура Ω. $\mathrm{At}(\Omega)$ — множество всех атомарных формул над Ω. Состояние — множество замкнутых атомарных формул.

Определение

Задача планирования STRIPS (‘**ST**anford **R**esearch **I**nstitute **P**roblem **S**olver’, 1971) — набор $\langle \mathcal{OP}, pre_+, pre_-, del, add, Init, Goal \rangle$, где

$\mathcal{OP}$ — конечное множество операций,

$pre_+, pre_-, del, add : \mathcal{OP} \to 2^{\mathrm{At}(\Omega)}$,

$Init, Goal \in 2^{\mathrm{At}(\Omega)}$ — начальное и заключительное состояния.

Определение

Выполнение операции $a \in \mathcal{OP}$ в состоянии s возможно, если существует подстановка θ, такая что $pre_+(a)\theta \subseteq s$ и $pre\_(a)\theta \cap s = \emptyset$. Результат выполнения — состояние $(s \setminus del(a)\theta) \cup add(a)\theta$.

План — последовательность операций $a_1 \dots a_n$, которые можно последовательно выполнить, начав с состояния $Init$, причем получающееся в итоге состояние содержит $Goal$.

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Общий случай (с функциональными символами): существование плана неразрешимо.

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Общий случай (с функциональными символами): существование плана неразрешимо.
- Без функциональных символов модель STRIPS — это не что иное как...

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Общий случай (с функциональными символами): существование плана неразрешимо.
- Без функциональных символов модель STRIPS — это не что иное как конечный автомат! Но с большим числом состояний.

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Общий случай (с функциональными символами): существование плана неразрешимо.
- Без функциональных символов

существование плана	EXPSPACE-полно
сущ. короткого плана	NEXPTIME-полно

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Общий случай (с функциональными символами): существование плана неразрешимо.
- Без функциональных символов и без del

существование плана	NEXPTIME-полно
сущ. короткого плана	NEXPTIME-полно

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Общий случай (с функциональными символами): существование плана неразрешимо.
- Без функциональных символов и без $del, pre\_$

существование плана	EXPTIME-полно
сущ. короткого плана	NEXPTIME-полно

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Пропозициональный случай (все предикаты 0-арные)

существование плана	PSPACE-полно
сущ. короткого плана	PSPACE-полно

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Пропозициональный случай (все предикаты 0-арные) без del

существование плана	NP-полно
сущ. короткого плана	NP-полно

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Пропозициональный случай (все предикаты 0-арные) без $del, pre\_$

существование плана	P
сущ. короткого плана	NP-полно

Алгоритмические задачи планирования

- Обычно исследуются такие алгоритмические задачи:
  - существование плана;
  - существование короткого плана: существует ли план длины не более k;
  - порождение плана: по задаче планирования выдать какой-нибудь план или ответить, что его нет;
  - верификация плана: проверить, является ли последовательность операций планом.
- Erol, Nau, Subrahmanian (1995). Complexity, decidability and undecidability results for domain-independent planning.
- Пропозициональный случай (все предикаты 0-арные) без $del, pre\_$

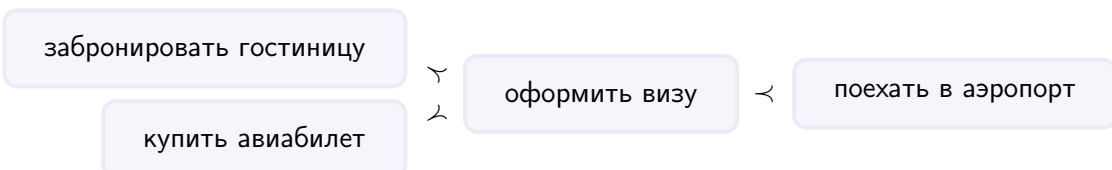
существование плана	P
сущ. короткого плана	NP-полно

- Из доклада Nicola Gigante:

  Complexity theory in a post-SAT-solvers world: NP-complete problems routinely solved (SAT, graph problems, scheduling) and even PSPACE is not that scary. Fear the worst, enjoy the average!

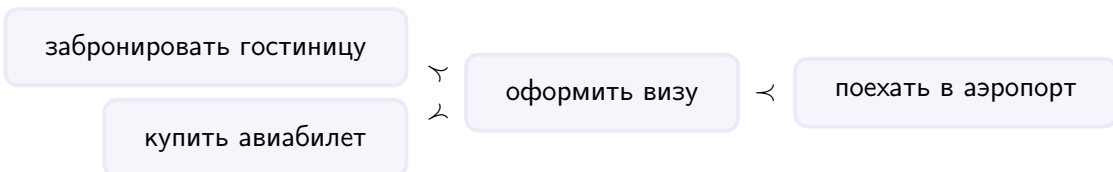
Иерархическое планирование

- Обычно, если нужно выполнить сложное задание, его разбивают на несколько более простых подзаданий.
- Задания удобно частично упорядочивать:



Иерархическое планирование

- Обычно, если нужно выполнить сложное задание, его разбивают на несколько более простых подзаданий.
- Задания удобно частично упорядочивать:



Даны множества $\mathcal{PT}$ имен примитивных заданий и $\mathcal{CT}$ имен составных заданий.

Определение

Сеть заданий — это набор $\langle T, \prec, \alpha \rangle$, где $\langle T, \prec \rangle$ — ч.у.м. и $\alpha : T \to \mathcal{PT} \cup \mathcal{CT}$.

Иерархическое планирование

Даны множества $\mathcal{PT}$ имен примитивных заданий и $\mathcal{CT}$ имен составных заданий.

Определение

Сеть заданий — это набор $\langle T, \prec, \alpha \rangle$, где $\langle T, \prec \rangle$ — ч.у.м. и $\alpha : T \to \mathcal{PT} \cup \mathcal{CT}$.

Декомпозиция задания $t \in T$ сетью заданий $\langle T', \prec', \alpha' \rangle$ — это сеть заданий $\langle \tilde{T}, \tilde{\prec}, \tilde{\alpha} \rangle$, получающаяся таким образом:

1. удалить t из T,
2. добавить свежую копию $\langle T', \prec', \alpha' \rangle$,
3. если $r \prec t$, то $r \mathrel{\tilde{\prec}} t'$ для всех $t' \in T'$,
4. если $t \prec r$, то $t' \mathrel{\tilde{\prec}} r$ для всех $t' \in T'$.

Пример

До декомпозиции: $t_1 \prec t_2 \prec t_3$.
Декомпозиция t_2 сетью заданий

$$u_1 \prec u_2 \begin{array}{l} \nearrow u_4 \\ \searrow u_3 \end{array}$$

После декомпозиции:

$$t_1 \prec u_1 \prec u_2 \begin{array}{l} \nearrow u_4 \\ \searrow u_3 \end{array} \begin{array}{l} \searrow \\ \nearrow \end{array} t_3$$

Иерархическое планирование

Даны множества $\mathcal{PT}$ имен примитивных заданий и $\mathcal{CT}$ имен составных заданий.

Определение

Сеть заданий — это набор $\langle T, \prec, \alpha \rangle$, где $\langle T, \prec \rangle$ — ч.у.м. и $\alpha : T \to \mathcal{PT} \cup \mathcal{CT}$.

Декомпозиция задания $t \in T$ сетью заданий $\langle T', \prec', \alpha' \rangle$ — это сеть заданий $\langle \tilde{T}, \tilde{\prec}, \tilde{\alpha} \rangle$, получающаяся таким образом:

1. удалить t из T,
2. добавить свежую копию $\langle T', \prec', \alpha' \rangle$,
3. если $r \prec t$, то $r \mathrel{\tilde{\prec}} t'$ для всех $t' \in T'$,
4. если $t \prec r$, то $t' \mathrel{\tilde{\prec}} r$ для всех $t' \in T'$.

Определение

Правило декомпозиции имеет вид $c \to tn$, где $c \in \mathcal{CT}$ — имя составного задания, а tn — сеть заданий.

Применение правила:

1. В сети заданий выбери задание t, такое что $\alpha(t) = c$.
2. Примени декомпозицию к t сетью tn.

Иерархическое планирование

Даны множества $\mathcal{PT}$ имен примитивных заданий и $\mathcal{CT}$ имен составных заданий.

Определение

Иерархическая задача планирования:
(1) конеч. набор правил декомпозиции,
(2) имя составного задания s и
(3) задача планирования STRIPS.

Решение $a_1 \dots a_n$ получается так:
(1) из s с помощью правил декомпозиции выведи сеть заданий tn с именами примитивных заданий, (2) возьми расширяющий ее линейный порядок $a_1 \prec \dots \prec a_n$, (3) проверь, что $a_1 \dots a_n$ является решением задачи STRIPS.

Определение

Правило декомпозиции имеет вид $c \to tn$, где $c \in \mathcal{CT}$ — имя составного задания, а tn — сеть заданий.
Применение правила:

1. В сети заданий выбери задание t, такое что $\alpha(t) = c$.
2. Примени декомпозицию к t сетью tn.

Иерархическое планирование

Даны множества $\mathcal{PT}$ имен примитивных заданий и $\mathcal{CT}$ имен составных заданий.

Определение

Иерархическая задача планирования:
(1) конеч. набор правил декомпозиции,
(2) имя составного задания s и
(3) задача планирования STRIPS.

Решение $a_1 \dots a_n$ получается так:
(1) из s с помощью правил декомпозиции выведи сеть заданий tn с именами примитивных заданий, (2) возьми расширяющий ее линейный порядок $a_1 \prec \dots \prec a_n$, (3) проверь, что $a_1 \dots a_n$ является решением задачи STRIPS.

- Методы декомпозиции обобщают контекстно-свободные грамматики

$$s \to a \prec s \prec b$$

- и коммутативные контекстно-свободные грамматики

$$s \to \begin{matrix} s \\ a \\ b \\ c \end{matrix}$$

Иерархическое планирование

Даны множества $\mathcal{PT}$ имен примитивных заданий и $\mathcal{CT}$ имен составных заданий.

Определение

Иерархическая задача планирования:
(1) конеч. набор правил декомпозиции,
(2) имя составного задания s и
(3) задача планирования STRIPS.

Решение $a_1 \dots a_n$ получается так:
(1) из s с помощью правил декомпозиции выведи сеть заданий tn с именами примитивных заданий, (2) возьми расширяющий ее линейный порядок $a_1 \prec \dots \prec a_n$, (3) проверь, что $a_1 \dots a_n$ является решением задачи STRIPS.

Erol, Hendler, Nau (1996). Complexity results for HTN planning.

- Задача существования плана **неразрешима** даже в пропозициональном случае!
- Но разрешима, если все сети заданий в правилах декомпозиции — линейные порядки.
  (Существование плана сводится к непустоте пересечения контекстно-свободного и регулярного языка.)

Пример: рецепт супа

Ингредиенты

- вода
- мясо, помытое и нарезанное кубиками
- овощи, помытые и нарезанные кубиками

Рецепт

1. Доведите воду до кипения.
2. Добавьте мясо и варите 1 час.
3. Добавьте овощи и варите еще 20 минут.

Пример: рецепт супа

Ингредиенты

- вода
- мясо, помытое и нарезанное кубиками
- овощи, помытые и нарезанные кубиками

Рецепт

1. Доведите воду до кипения.
2. Добавьте мясо и варите 1 час.
3. Добавьте овощи и варите еще 20 минут.

Формализация

$$\texttt{soup} \to \texttt{water} \prec \texttt{meat} \prec \texttt{veg}$$

$$\texttt{meat} \to \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat})$$

$$\texttt{veg} \to \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

Пример: рецепт супа

Формализация

$$\texttt{soup} \to \texttt{water} \prec \texttt{meat} \prec \texttt{veg}$$

$$\texttt{meat} \to \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat})$$

$$\texttt{veg} \to \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

$$\texttt{soup}$$

Пример: рецепт супа

Формализация

$$\texttt{soup} \to \texttt{water} \prec \texttt{meat} \prec \texttt{veg}$$

$$\texttt{meat} \to \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat})$$

$$\texttt{veg} \to \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

$$\texttt{water} \prec \texttt{meat} \prec \texttt{veg}$$

Пример: рецепт супа

Формализация

$$\texttt{soup} \to \texttt{water} \prec \texttt{meat} \prec \texttt{veg}$$

$$\texttt{meat} \to \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat})$$

$$\texttt{veg} \to \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

$$\texttt{water} \prec \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat}) \prec \texttt{veg}$$

Пример: рецепт супа

Формализация

$$\texttt{soup} \to \texttt{water} \prec \texttt{meat} \prec \texttt{veg}$$

$$\texttt{meat} \to \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat})$$

$$\texttt{veg} \to \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

$$\texttt{water} \prec \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat}) \prec \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

Пример: рецепт супа

Формализация

$$\texttt{soup} \to \texttt{water} \prec \texttt{meat} \prec \texttt{veg}$$
$$\texttt{meat} \to \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat})$$
$$\texttt{veg} \to \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

$$\texttt{water} \prec \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat}) \prec \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

Но секунду...

Получается, мыть и резать овощи можно только после того, как отварил мясо?!

Пример: рецепт супа

Формализация

$$\begin{aligned} \texttt{soup} &\to \texttt{water} \prec \texttt{meat} \prec \texttt{veg} \\ \texttt{meat} &\to \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat}) \\ \texttt{veg} &\to \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg}) \end{aligned}$$

$$\texttt{water} \prec \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat}) \prec \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

Но секунду...

Получается, мыть и резать овощи можно только после того, как отварил мясо?!

Чтобы исправить ситуацию, можно

A. придумать другие правила декомпозиции,

B. задействовать STRIPS или

Пример: рецепт супа

Формализация

$$\texttt{soup} \to \texttt{water} \prec \texttt{meat} \prec \texttt{veg}$$
$$\texttt{meat} \to \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat})$$
$$\texttt{veg} \to \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

$$\texttt{water} \prec \texttt{wash}(\texttt{meat}) \prec \texttt{cut}(\texttt{meat}) \prec \texttt{smr}(\texttt{meat}) \prec \texttt{wash}(\texttt{veg}) \prec \texttt{cut}(\texttt{veg}) \prec \texttt{smr}(\texttt{veg})$$

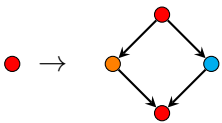
Но секунду...

Получается, мыть и резать овощи можно только после того, как отварил мясо?!

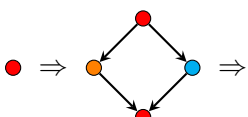
Чтобы исправить ситуацию, можно

A. придумать другие правила декомпозиции,
B. задействовать STRIPS или
C. сделать правила декомпозиции более гибкими.

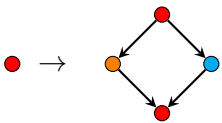
Грамматики замещения вершин: пример

Правило: 

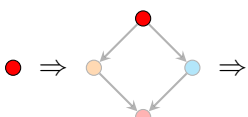
Вывод:



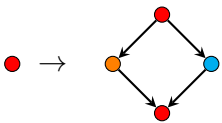
Грамматики замещения вершин: пример

Правило: 

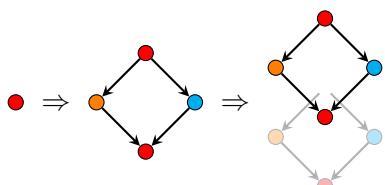
Вывод:



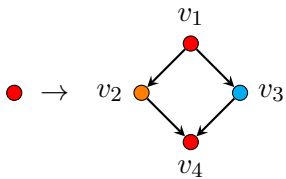
Грамматики замещения вершин: пример

Правило: 

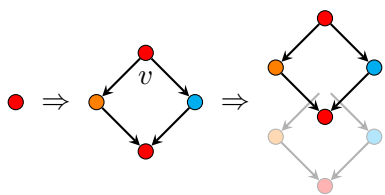
Вывод:



Грамматики замещения вершин: пример

Правило: 

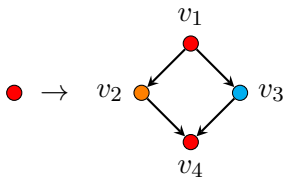
Вывод:



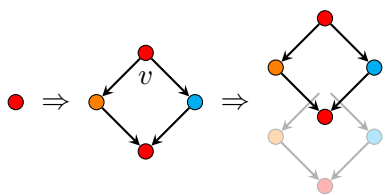
Инструкции по соединению

(1) Если до применения правила было ребро из v в оранжевую вершину, создай ребра из v_2 и v_4 в эту оранжевую вершину.

Грамматики замещения вершин: пример

Правило: 

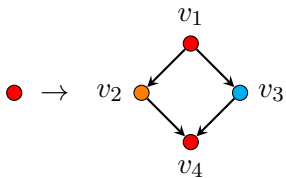
Вывод:



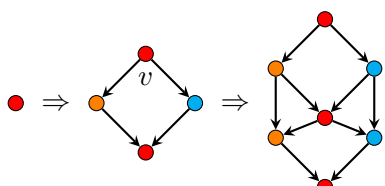
Инструкции по соединению

(2) Если до применения правила было ребро из v в голубую вершину, создай ребра из v_3 и v_4 в эту голубую вершину.

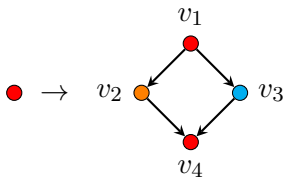
Грамматики замещения вершин: пример

Правило: 

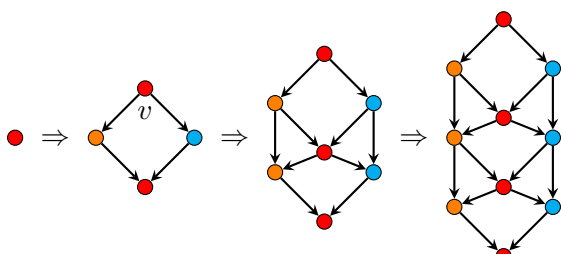
Вывод:



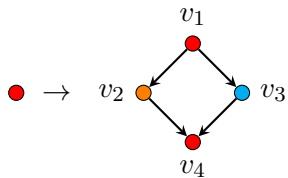
Грамматики замещения вершин: пример

Правило: 

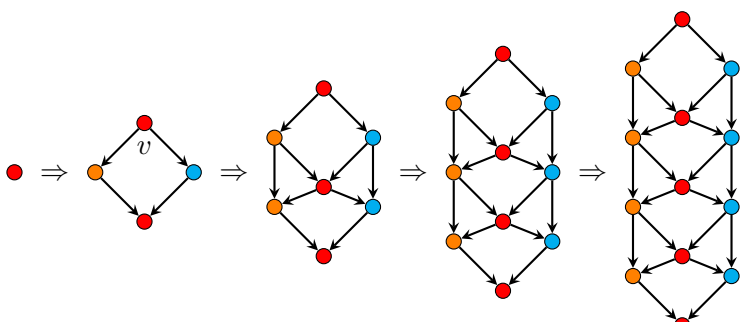
Вывод:



Грамматики замещения вершин: пример

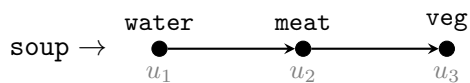
Правило: 

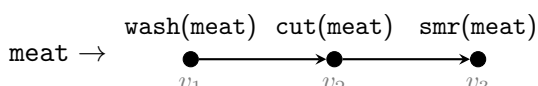
Вывод:



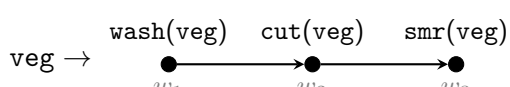
Иерархическое планирование и графовые грамматики

- Вместо правил декомпозиции в иерархическом планировании можно использовать правила грамматик замещения вершин.
- Это, например, позволит исправить формализацию супа:





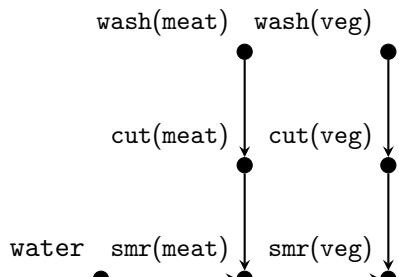
+инструкция: создавай новые ребра только к и от v_3



+инструкция: создавай новые ребра только к и от w_3

Иерархическое планирование и графовые грамматики

- Вместо правил декомпозиции в иерархическом планировании можно использовать правила грамматик замещения вершин.
- В данной грамматике из `soup` выводится граф **частичного** порядка



Планирование и формальные языки

- Множество планов для данной задачи планирования (STRIPS/иерархической/графово-грамматической) образует формальный язык.
- Философия: множество планов показывает, насколько сложно устроены механизмы построения планов в формализме, и позволяет сравнивать их между собой.

Планирование и формальные языки

- Множество планов для данной задачи планирования (STRIPS/иерархической/графово-грамматической) образует формальный язык.
- Философия: множество планов показывает, насколько сложно устроены механизмы построения планов в формализме, и позволяет сравнивать их между собой.
- Таким образом, тип задач планирования определяет класс формальных языков.

пропозициональный STRIPS	регулярные языки
иерархическое планирование	???
иерарх. план. с линейными порядками	контекстно-свободные языки
грамматики замещения вершин	$\supseteq$ множественные кс-языки

Планирование и формальные языки

- Множество планов для данной задачи планирования (STRIPS/иерархической/графово-грамматической) образует формальный язык.
- Философия: множество планов показывает, насколько сложно устроены механизмы построения планов в формализме, и позволяет сравнивать их между собой.
- Таким образом, тип задач планирования определяет класс формальных языков.

пропозициональный STRIPS	регулярные языки
иерархическое планирование	???
иерарх. план. с линейными порядками	контекстно-свободные языки
грамматики замещения вершин	$\supseteq$ множественные кс-языки

- Гипотеза: язык $\{ww \mid w \in \{a, b\}^*\}$ не является множеством планов ни одной задачи иерархического планирования.