Значимый контекст рассуждений как результат абстрагирования

Трофимов И.В.

Известно, что задача планирования в общем виде является NP-задачей [Bylander, 1991] [Bylander, 1994] [Erol et al., 1991]. Довольно часто в практике планирования в качестве метода борьбы со сложностью выбирают абстрагирование.

1 Абстрагирование в искусственном интеллекте

> Абстракция — мысленное отвлечение, обособление от тех или иных сторон, свойств или связей предметов и явлений для выделения существенных их признаков.
>
> (толковый словарь Ожегова – Шведовой)

Традиционно в литературе по искусственному интеллекту *абстрагирование* определяется как частный случай *изменения способа представления* (representation change). Как отмечают современные исследователи проблемы абстрагирования [Holte, Choueiry, 2003], сложно дать понятию «абстрагирование» точную формулировку, которая была бы достаточно общей, чтобы включать все разнообразие методов (технологий), *интуитивно* попадающих под категорию «абстрагирование», и в то же время была бы достаточно специфичной, чтобы исключить другие формы изменения способа представления, которые интуитивно абстрагированием не являются (например, реформулировку — reformulation[1]). Ниже приводится два нестрогих (интуитивных) определения понятия абстрагирование, отражающие современное представление о нем.

> Абстракция — это отображение из одного представления задачи в другое, обеспечивающее определенные желаемые свойства[2] и уменьшающее сложность.
>
> [Giunchiglia et al., 1997] со ссылкой на [Giunchiglia, Walsh, 1992]

[1] Некоторые исследователи, например [Holte, Choueiry, 2003], считают *реформулировку* (reformulation) и *изменение способа представления* (representation change) синонимами. Другие исследователи, например [Zucker, 2003], предлагают рассматривать *реформулировку* как частный случай *изменения способа представления*, предполагающий переход к новому формализму без потери информации. Примером реформулировки может служить переход от декартовой к полярной системе координат. У Зюкера (Zucker) *реформулировка* противопоставляется *абстрагированию*; последнее определяется как вид *изменения способа представления* в рамках одного формализма, причем уменьшающее количество информации. По вопросу определения понятий *реформулировка* и *изменение способа представления* автор разделяет взгляды Зюкера.

[2] В оригинале — preserve certain desirable properties. В планировании примерами желаемых свойств (согласно [Zucker, 2003]) могут служить *свойство монотонности* или *свойство нисходящего уточнения* [Knoblock, 1991]. В общем случае, свойства зависят от области, где применяется абстрагирование.

> Абстракция — это изменение способа представления в рамках одного и того же формализма, которое скрывает несущественные детали и обеспечивает желаемые свойства.
>
> [Zucker, 2003]

В последнем определении отмечается, что абстрагирование само по себе не предполагает смены формализма (смена формализма — это реформулировка). Однако абстрагирование и реформулировка иногда комбинируются в рамках одного процесса изменения способа представления, и такой процесс «по инерции» (и, видимо, ошибочно) называют абстрагированием.

Методы абстрагирования можно разделить на два класса по механизму, который используется для скрытия несущественных деталей. По мнению автора, таких механизмов два (существуют и другие классификации).

1. *Скрытие элементов предметной области*. В качестве элементов могут выступать объекты, процессы, явления, действия и так далее. В литературе такой механизм называют фокусировкой (focusing, focalization) или скрытием домена (domain hiding) [Zucker, 2003] или проекцией (projection) [Holte, Choueiry, 2003].
2. *Скрытие свойств элементов предметной области*. Скрытие свойств может сделать отдельные элементы домена неразличимыми (уменьшается разнообразие), но число элементов не изменяется. Скрытие свойств может осуществляться не для всех элементов домена, а только для определенных групп или даже отдельных элементов. В литературе этот механизм называют также «избирательность» (selectivity) или скрытием содомена (co-domain hiding) [Zucker, 2003] или абстрагированием домена (domain abstraction) [Holte, Choueiry, 2003].

В [Zucker, 2003] приводится еще ряд механизмов абстрагирования, но автор не разделяет такую позицию. Domain reduction и co-domain reduction в действительности являются аппроксимацией, так как приводят к появлению новых сущностей (в первом случае) или трансформации существующих (во втором), а не скрытию элементов или свойств. Domain aggregation является частным случаем фокусировки. Последнее утверждение можно обосновать следующим образом.

Агрегирование, и противоположную ему декомпозицию[3], при определенных обстоятельствах можно рассматривать как частные случаи абстрагирования, механизмом

[3] Разумеется, в данном контексте под «агрегированием» или «декомпозицией» подразумевается ментальный процесс, а не физический.

которого является фокусировка. Действительно, оба эти термина относятся к операциям с системами. Когда мы имеем дело с системой, то подразумевается, что имеет место как сама система (как некоторая целостность), так и части (и их взаимосвязи), из которых она состоит. Когда мы осуществляем агрегирование, то абстрагируемся от *частей* системы (как элементов предметной области), то есть фокусируемся на самой системе и ее свойствах. Наоборот, когда мы осуществляем декомпозицию, мы абстрагируемся от самой системы (целого), и фокусируемся только на ее частях и связях между ними. Однако не всегда агрегирование и декомпозиция являются видами абстрагирования. Если в представлении (representation) понятие системы до агрегирования не существовало, то имеет место не скрытие частей (абстрагирование), а порождение нового элемента в представлении (системы). И это уже не абстрагирование. Неточности такого рода иногда встречаются в литературе; например в [Holte et al., 1994] [Holte et al., 1996] абстрагированием называлась группировка (даже не агрегирование) состояний пространства поиска с дальнейшим замещением группы на одно новое «абстрактное» состояние. В действительности, при порождении «абстрактного пространства поиска» абстрагирование имело место лишь в момент скрытия связей между элементами группы; все остальное — синтез и преобразования иного рода.

2 Абстрагирование в планировании

Существует множество планировщиков и теоретических исследований в области планирования, авторы которых пытались использовать абстрагирование, как способ преодоления проблемы вычислительной сложности. Во всех случаях можно выделить общую методологическую основу.

1. Сначала осуществляется переход к абстрактной формулировке задачи.
2. Затем решается абстрактная задача, сложность которой меньше сложности исходной задачи.
3. И, наконец, на базе абстрактного решения отыскивается решение исходной задачи.

Абстрагирование формулировки задачи в большинстве работ сводится к синтаксическому удалению элементов предусловия действий (типично, элементов конъюнкции в предусловиях, выраженных в форме конъюнкции литералов). К числу таких планировщиков относятся системы ABSTRIPS [Sacerdoti, 1974], ABTWEAK [Yang, Tenenberg, 1990], PRODIGY/ALPINE [Knoblock, 1994]. Абстрагирование можно осуществлять также путем синтаксического удаления элементов эффекта действий, например, в [Fink, Yang, 1992], а также в системах HSP [Bonet, Geffner, 1998] и FF [Hoffmann, Nebel, 2001]. Оригинальный подход с использованием абстрагирования был

предложен в [Holte et al., 1994] [Holte et al., 1996]; абстрагирование использовалось для построения пространств поиска меньшего размера, обладающих свойством нисходящего уточнения (или монотонности уточнения). Задача планирования формулировалась в виде графа пространства поиска, а абстрагирование сводилось к отвлечению от ряда дуг в графе в процессе построения нового редуцированного пространства поиска.

Что касается использования абстрактного решения, то здесь существуют два основных метода. Первый заключается в том, что длина абстрактного решения используется как эвристическая оценка расстояния до цели. На базе этого метода реализованы планировщики HSP и FF. Во втором методе само абстрактное решение выступает в качестве «каркаса» (скелета) искомого решения, и необходимо найти лишь подходящую его конкретизацию. На базе этого метода реализованы планировщики ABSTRIPS, ABTWEAK, PRODIGY/ALPINE. Важным условием здесь является возможность создания эффективного алгоритма конкретизации абстрактного решения. При определенных ограничениях это условие выполняется. Однако если оно не выполняется, то такой подход может оказаться гораздо менее эффективным, чем методы, не использующие абстрагирование [Bacchus, Yang, 1994].

Все упомянутые в этом разделе планировщики (за исключением решения, предложенного в [Holte et al., 1994]) осуществляют абстрагирование путем скрытия *свойств* элементов предметной области, а именно, свойств действий (или схем действий). Однако не менее полезным для поиска решения проблемы вычислительной сложности является скрытие самих *элементов* предметной области (фокусировка), а именно, объектов и действий, которые не могут иметь отношение к решению текущей задачи. Отсев нерелевантных объектов и действий в значительной мере сокращает пространство поиска. Кроме того, важным свойством фокусировки является то, что *абстрактное решение является также и конкретным решением*, таким образом, нет необходимости осуществлять конкретизацию. Автору не известны планировщики, осуществляющие абстрагирование таким путем.

3 Модель фокусировки для задачи планирования

Для того чтобы рассуждать о том, какие объекты и действия имеют или не имеют отношение к решению текущей задачи планирования, необходимо определить ряд вспомогательных понятий. Эти определения опираются на определение задачи планирования. Здесь будет использоваться упрощенная формулировка задачи планирования, так как ряд аспектов, обычно задействованных в классических способах формулировки (например, PDDL), в рамках данной работы интереса не представляют и приводят лишь к ненужным усложнениям.

Итак, при решении задачи планирования предполагается наличие *агента*, способного на некоторую деятельность, и *внешней среды*, в которой агент действует. Агент способен выполнять конечное количество видов деятельности, описываемых *схемами действий*. Схема действий — это параметризованное описание некоторого вида деятельности; подстановка имен конкретных объектов среды вместо параметров порождает определение конкретного действия.

Обозначим **Schemas** — конечное множество схем действий, описывающих все возможные виды деятельности для данного агента. Каждому элементу множества **Schemas** соответствует целое неотрицательное число **k**, обозначающее количество объектов, вовлеченных в описание действий, соответствующих данной схеме. Будем называть это число *арностью схемы действий* или *количеством параметров схемы действий*. Арность схемы действий удобно моделировать при помощи функции:

μ: **Schemas** $\rightarrow \mathbf{Z}^{+}$.

При письме для обозначения арности схемы действий будет использоваться верхний индекс. Например, запись $\mathbf{schema}^{\mathbf{k}}$ означает, что речь идет о такой **schema**, для которой $\mu(\mathbf{schema}) = \mathbf{k}$.

Каждому описанию схемы действий соответствует непустое множество описаний конкретных действий. Описание действия представляет собой кортеж вида: $\langle \mathbf{schema}^{\mathbf{k}}, \mathbf{obj}_1, \ldots, \mathbf{obj}_{\mathbf{k}} \rangle$, где $\mathbf{obj}_{\mathbf{i}}$ — имена объектов, вовлеченных в описание действия. Способы формирования описаний конкретных действий из схем могут быть различны и здесь не рассматриваются. Множество всех действий, соответствующих некоторой схеме действий **schema**, будем обозначать **actions**[**schema**], а множество всех возможных действий агента обозначим **Actions**.

$$\mathbf{Actions} = \bigcup_{\mathbf{schema} \in \mathbf{Schemas}} \mathbf{actions}[\mathbf{schema}]$$

Во всякий момент времени внешняя среда находится в каком-либо состоянии. Множество всех возможных состояний, в которых может пребывать внешняя среда, обозначим **States**. Для каждого состояния в **States** определено множество действий из **Actions**, которые выполнимы в данном состоянии. Другими словами существует функция

γ: **Actions** $\times$ **States** $\rightarrow$ {*выполнимо*, *невыполнимо*},

которая для каждого действия из **Actions** и каждого состояния среды из **States** определяет, выполнимо ли действие в данном состоянии. Функцию γ будем называть *функцией выполнимости*.

Выполнение всякого действия приводит к изменению состояния среды. Обозначим **AS** множество всех пар ⟨**action**, **state**⟩ (**action**∈**Actions**, **state**∈**States**) для которых выполняется условие (**x**∈**AS** → (**γ**(**x**) = *выполнимо*)). Определим функцию

φ: **AS** → **States**,

причем если **φ**(**action**, $\mathbf{state_1}$)=$\mathbf{state_2}$, то $\mathbf{state_1}\neq\mathbf{state_2}$. Функцию **φ** будем называть *функцией переходов*.

Определим *задачу планирования* как кортеж вида:

⟨**States**, **Actions**, **γ**, **φ**, $\mathbf{S_0}$, **SG**⟩,

где $\mathbf{S_0}$∈**States** — состояние, называемое *начальным*; **SG**⊂**States** — множество состояний, являющихся *целевыми* для данной задачи планирования (в этих состояниях выполняется целевое условие). $\mathbf{S_0}\notin$**SG**.

Решением задачи планирования (или *планом*) будем называть последовательность[4] вида

⟨$\mathbf{S_0}$, $\mathbf{A_0}$, $\mathbf{S_1}$, $\mathbf{A_1}$, …, $\mathbf{S_N}$, $\mathbf{A_N}$, $\mathbf{S_{N+1}}$⟩,

для которой выполняются следующие условия:

1. $\mathbf{S_i}\in$**States**, $\mathbf{A_j}\in$**Actions**, где $0 \le \mathbf{i} \le \mathbf{N+1}$, $0 \le \mathbf{j} \le \mathbf{N}$;
2. $\mathbf{S_{N+1}}\in$**SG**;
3. $\mathbf{S_i}\notin$**SG**, где $0 \le \mathbf{i} \le \mathbf{N}$;
4. **γ**($\mathbf{A_i}$, $\mathbf{S_i}$)=*выполнимо*, где $0 \le \mathbf{i} \le \mathbf{N}$;
5. **φ**($\mathbf{A_i}$, $\mathbf{S_i}$)=$\mathbf{S_{i+1}}$, где $0 \le \mathbf{i} \le \mathbf{N}$.

Пусть **P** — некоторая задача планирования, тогда **Solutions**(**P**) будем обозначать множество всех возможных решений задачи планирования **P**.

Определив таким образом постановку задачи планирования и результат ее решения, перейдем к определению понятий, характеризующих различные виды фокусировки (как разновидности абстрагирования) для задачи планирования.

Для данной задачи **P** = ⟨**States**, **Actions**, **γ**, **φ**, $\mathbf{S_0}$, **SG**⟩ *результатом фокусировки* будет являться абстрактная задача $\mathbf{P_1}$ = ⟨$\mathbf{States_1}$, $\mathbf{Actions_1}$, **γ**, **φ**, $\mathbf{S_0}$, $\mathbf{SG_1}$⟩, отвечающая следующим требованиям:

1. $\mathbf{States_1}\subseteq$**States**;
2. $\mathbf{Actions_1}\subseteq$**Actions**;
3. **a**∈$\mathbf{Actions_1}$, **s**∈$\mathbf{States_1}$, **γ**(**a**, **s**)=*выполнимо* → **φ**(**a**,**s**)∈$\mathbf{States_1}$;

[4] В данной работе для простоты рассматриваются только планы без параллельных ветвей деятельности (линеаризованные планы).

4. $\mathbf{SG_1} \subset \mathbf{States_1}$;
5. $\mathbf{SG_1} \subseteq \mathbf{SG}$;
6. $\mathbf{Solutions}(\mathbf{P_1}) \subseteq \mathbf{Solutions}(\mathbf{P})$;
7. $\mathbf{Solutions}(\mathbf{P}) \neq 0 \rightarrow \mathbf{Solutions}(\mathbf{P_1}) \neq 0$.

Налагая дополнительные ограничения на $\mathbf{P_1}$ можно определить несколько разновидностей фокусировки.

Если для $\mathbf{P_1}$ выполняется условие $\mathbf{Solutions}(\mathbf{P_1}) = \mathbf{Solutions}(\mathbf{P})$, то фокусировку будем называть *полно-сохраняющей*. Множество $\mathbf{Actions_1}$ в этом случае будем называть *полным значимым контекстом*, а множество $\mathbf{GNCx} = \mathbf{Actions} \setminus \mathbf{Actions_1}$, *гарантированно-незначимым контекстом*.

Если для $\mathbf{P_1}$ выполняется условие $\mathbf{SG_1} = \mathbf{SG}$, и для каждого $\mathbf{s} \in \mathbf{SG_1}$ существует хотя бы одно $\mathbf{sol} \in \mathbf{Solutions}(\mathbf{P_1})$, последним элементом которого является $\mathbf{s}$, то такую фокусировку будем называть *целе-сохраняющей*. Множество $\mathbf{Actions_1}$ в этом случае будем называть *целе-полным значимым контекстом*.

Если для $\mathbf{P_1}$ выполняется условие $|\mathbf{Solutions}(\mathbf{P_1})| = 1$ (при $\mathbf{Solutions}(\mathbf{P}) \neq 0$), то фокусировку будем называть *минималистской*. Множество $\mathbf{Actions_1}$ в этом случае будем называть *минимальным значимым контекстом*.

Если для $\mathbf{P_1}$ выполняется условие $\mathbf{Actions_1} \subset \mathbf{Actions}$, то фокусировку будем называть *полезной*, а множество $\mathbf{Actions_1}$ в этом случае будем называть *полезным значимым контекстом*.

Полно-сохраняющие фокусировки сохраняют возможность достижения любого из целевых состояний всеми возможными способами. Это имеет значение в случае, если нам необходимо найти все планы или план, на который налагается оптимизационное условие (например, найти план минимальной длины или минимальный по времени исполнения). Целе-сохраняющие фокусировки сохраняют лишь возможность достижения любого из целевых состояний. Это важно, если на целевое состояние налагается дополнительное условие, которое невозможно выразить через саму цель (например, оптимизационное «максимизировать количество оставшегося топлива). Если же необходимости достижения любого из целевых состояний нет, то для сокращения пространства поиска желательно найти минималистскую фокусировку (если она существует) или полезную фокусировку с множеством $\mathbf{Actions_1}$ как можно меньшей мощности.

4 Механизм фокусировки

Предложенная модель фокусировки предполагает, что абстрактная задача отличается от исходной содержанием трех множеств: множества состояний, множества действий и множества целевых состояний. Или более конкретно, в абстрактной задаче эти множества являются подмножествами соответствующих множеств исходной задачи или эквивалентны им. В данной работе рассматривается механизм фокусировки, сокращающий *только* множество **Actions**, то есть предполагается выполнение следующих условий $\mathbf{States_1} = \mathbf{States}$, $\mathbf{SG_1} = \mathbf{SG}$ и $\mathbf{Actions_1} \subset \mathbf{Actions}$. По ранее предложенной классификации такая фокусировка является, по меньшей мере, полезной.

Абстрагирование множества действий (путем взятия его подмножества) приводит к тому, что некоторые состояния (в том числе и целевые) могут стать недостижимыми. Таким образом, побочным эффектом такого способа абстрагирования является и возможное сокращение множества состояний (целевых состояний). Однако в явном виде над множествами **States** и **SG** не будет производиться никаких операций в процессе абстрагирования.

Так как множества **States** и **SG** при фокусировке остаются неизменными, требования 1, 3, 4 и 5 к результату фокусировки выполняются автоматически. Требование 2 выполняется в силу определения механизма абстрагирования — $\mathbf{Actions_1} \subset \mathbf{Actions}$.

Требование 6 выполняется в силу следующих обстоятельств. По определению **Solutions**(**P**) содержит все возможные решения исходной задачи **P**. Так как функции выполнимости (**γ**) и переходов (**φ**) остаются неизменными в процессе абстрагирования, новые решения не могут быть получены. А так как множество действий в результате абстрагирования сокращается, некоторые решения перестанут существовать (из-за отсутствия содержавшегося в них действия или нескольких действий). Поэтому требование 6 справедливо.

Таким образом, при абстрагировании путем сокращения множества **Actions** необходимо заботиться лишь о выполнении требования 7, которое постулирует следующее. Если имеется хотя бы одно решение исходной задачи, то абстрактная задача также должна иметь хотя бы одно решение. Это условие автоматически выполняется, только если из множества **Actions** исключается гарантированно-незначимый контекст **GNCx** или его подмножество.

С учетом требования 7, если существует решение исходной задачи ($\mathbf{Solutions}(\mathbf{P}) \neq 0$), то для выполнения фокусировки множество $\mathbf{Actions_1}$ должно содержать набор действий, из которых можно получить хотя бы одно решение задачи. Это

абстрактное множество действий будем называть *возможным значимым контекстом* и обозначать **SCx**. Если для получения решения нельзя исключить ни одного действия, то абстрагирование невозможно. Исключенные (абстрагированные) действия составляют множество, которое будем называть *допустимым незначимым контекстом* и обозначать **NCx**. Эти два вида контекстов связаны следующим соотношением.

SCx = **Actions** \ **NCx**

Таким образом, для того чтобы осуществить абстрагирование, нам достаточно найти либо один из возможных значимых контекстов, либо какой-нибудь допустимый незначимый контекст.

Чтобы воспользоваться предложенным механизмом абстрагирования необходимо решить две подзадачи:

- найти адекватное *средство определения* для возможного значимого или допустимого незначимого контекста;
- найти *механизм выявления* **NCx** или **SCx** для текущей задачи планирования.

5 Способы определения возможного значимого или допустимого незначимого контекстов

По определению возможный значимый и допустимый незначимый контексты (далее в этом разделе, просто контексты) являются множествами. Два основных способа определения множеств — перечисление элементов и определение посредством характеристического свойства.

В первом случае контекст представляет собой простое перечисление действий. Это удобно тем, что для осуществления планирования сразу имеется готовое множество действий, которыми можно оперировать. С другой стороны, это множество может быть огромной мощности и потребует больших затрат на хранение. В открытых средах, где количество объектов (и оперирующих ими конкретных действий) не определено и считается бесконечным, такой подход к определению контекста становится неприемлемым. То же самое касается сред, в которых параметрами действий могут выступать числовые величины (например, вещественные числа или целые из неограниченных интервалов).

Второй способ позволяет иметь дело с бесконечными множествами (действий). Однако в процессе планирования при выборе очередного действия необходимо выполнять проверку на предмет удовлетворения этого действия ограничениям характеристического свойства, чтобы выяснить, принадлежит ли действие контексту. А это требует некоторых вычислительных ресурсов.

В общем виде, определение контекста **Cx** посредством характеристического свойства можно записать следующим образом:

$$\mathbf{Cx} = \{\mathbf{a} \mid \mathbf{a} \in \mathbf{Actions}, \mathbf{filter}\},$$

где **filter** —требование к элементам **Cx** (характеристическое свойство). То есть контекст представляет собой такое множество элементов из **Actions**, которые отвечают требованию **filter**. Рассмотрим подробнее, какими могут быть требования.

В силу того, что мы моделируем действие как составную сущность, а именно, как кортеж вида $\langle \mathbf{schema^k}, \mathbf{obj_1}, \ldots, \mathbf{obj_k} \rangle$, мы можем накладывать ограничения на элементы, из которых состоит действие.

Во-первых, можно потребовать, чтобы элементами контекста были действия, порожденные от одной определенной схемы действий. Это можно сформулировать следующим образом.

$$\mathbf{Cx} = \{\mathbf{a} \mid \mathbf{a} \in \mathbf{Actions}, \mathbf{a} \in \mathbf{actions}[\mathbf{schema}]\}, \text{ где } \mathbf{schema} \in \mathbf{Schemas}.$$

Во-вторых, требование может постулировать, что **n**-ым параметром действия является имя конкретного объекта **Obj**. Это также удобно определить посредством множества. Обозначим $\mathbf{actions_{i,Obj}}[\mathbf{schema^k}]$ множество действий, порожденных от схемы $\mathbf{schema^k}$, у которых в качестве **i**-ого параметра (и, следовательно, **i+1**-ого элемента кортежа, описывающего действие) выступает имя объекта **Obj** (здесь $1 \le \mathbf{i} \le \mathbf{k}$). Тогда множеством всех действий, у которых **i**-ый параметр равен **Obj**, является объединение множеств $\mathbf{actions_{i,Obj}}[\mathbf{schema^k}]$ по всем схемам действий с арностью большей либо равной **i**. Будем обозначать это множество $\mathbf{actions_{i,Obj}}$.

$$\mathbf{actions_{i,Obj}} = \bigcup_{\substack{\mathbf{schema^k} \in \mathbf{Schemas}, \\ k \ge i}} \mathbf{actions_{i,Obj}}[\mathbf{schema^k}], \qquad (\mathbf{i} \ge 1)$$

Соответственно, контекст будет определяться как

$$\mathbf{Cx} = \{\mathbf{a} \mid \mathbf{a} \in \mathbf{Actions}, \mathbf{a} \in \mathbf{actions_{i,Obj}}\}.$$

Разумеется, в качестве требования может выступать и принадлежность или не принадлежность **a** множеству $\mathbf{actions_{i,Obj}}[\mathbf{schema^k}]$.

В-третьих, рассуждая аналогичным образом, можно сформулировать требование, согласно которому контекст состоит из действий, оперирующих объектом **Obj**. Множество таких действий будем обозначать $\mathbf{actions_{Obj}}$.

$$\mathbf{actions_{Obj}} = \bigcup_{\substack{\forall i, \\ (i \ge 1)}} \mathbf{actions_{i,Obj}}$$

Определение соответствующего контекста будет выглядеть так

$$\mathbf{Cx} = \{\mathbf{a} \mid \mathbf{a} \in \mathbf{Actions}, \mathbf{a} \in \mathbf{actions}_{\mathbf{Obj}}\}.$$

Такой контекст определяет, что все действия, составляющие контекст, должны оперировать объектом **Obj**. Для возможного значимого контекста требование непринадлежности такому множеству фактически исключает объект из рассмотрения. То есть требование

$$\mathbf{SCx} = \{\mathbf{a} \mid \mathbf{a} \in \mathbf{Actions}, \mathbf{a} \notin \mathbf{actions}_{\mathbf{Obj}}\}$$

в действительности постулирует абстрагирование от одного определенного объекта предметной области (посредством абстрагирования от операций с ним).

И, наконец, в-четвертых, требование может ограничить контекст действиями, оперирующими объектом **Obj** и, кроме того, порожденными от одной конкретной схемы (а не от любой, как в предыдущем случае). Множество таких действий обозначим $\mathbf{actions}_{\mathbf{Obj}}[\mathbf{schema}^{\mathbf{k}}]$.

$$\mathbf{actions}_{\mathbf{Obj}}[\mathbf{schema}^{\mathbf{k}}] = \bigcup_{\substack{\forall i, \\ (1 \le i \le k)}} \mathbf{actions}_{\mathbf{i,Obj}}[\mathbf{schema}^{\mathbf{k}}]$$

А соответствующее определение контекста будет иметь вид

$$\mathbf{Cx} = \{\mathbf{a} \mid \mathbf{a} \in \mathbf{Actions}, \mathbf{a} \in \mathbf{actions}_{\mathbf{Obj}}[\mathbf{schema}^{\mathbf{k}}]\}.$$

Так как мы задаем характеристическое свойство посредством операции принадлежности или непринадлежности какому-либо множеству, мы можем строить из предложенных выше множеств ($\mathbf{actions}[\mathbf{schema}]$, $\mathbf{actions}_{\mathbf{i,Obj}}[\mathbf{schema}^{\mathbf{k}}]$, $\mathbf{actions}_{\mathbf{i,Obj}}$, $\mathbf{actions}_{\mathbf{Obj}}[\mathbf{schema}^{\mathbf{k}}]$ и $\mathbf{actions}_{\mathbf{Obj}}$), а также из индивидуальных действий, более сложные требования к контексту, используя операции над множествами. Например, нам может потребоваться контекст, исключающий из рассмотрения объекты **box** и **block**, причем состоящий только из действий, порожденных от схем **schema-one** и **schema-four**. Такой контекст можно определить следующим образом.

$$\mathbf{Cx} = \{\mathbf{a} \mid \mathbf{a} \in \mathbf{Actions},$$
$$\mathbf{a} \in ((\mathbf{actions}[\mathbf{schema\text{-}one}] \cup \mathbf{actions}[\mathbf{schema\text{-}four}]) \setminus \mathbf{actions}_{\mathbf{box}} \setminus \mathbf{actions}_{\mathbf{block}})\}$$

Способ задания контекста посредством характеристического свойства довольно гибок. Переходя к более сложным моделям задачи планирования, можно составлять характеристические требования другого рода — оперирующие понятиями новой модели. Например, если рассматривать действия как сущности обладающие предусловием и эффектом (именно так обычно определяют действия теоретики автоматического планирования), то соответственно можно придумать и ввести ограничения на эти элементы. Кроме этого, оперируя с объектами (не как с набором имен, а как с более

сложными сущностями) можно осуществлять их отбор согласно набору их характеристик. В данной работе такие подходы не рассматриваются.

Задание контекста посредством характеристического свойства «включает в себя» выразительные способности задания контекста путем перечисления. Можно составить «фильтрующее множество», представляющее собой объединение индивидуальных действий, входящих в перечисление, и затем использовать проверку на принадлежность этому множеству в качестве характеристического свойства.

6 Способы выявления возможного значимого или допустимого незначимого контекстов для определенной задачи планирования

Методы выявления контекстов могут опираться на информацию о:

- задачах планирования в целом;
- классах задач планирования;
- текущей решаемой задаче планирования.

Чем более общую информацию использует метод, тем он более универсальный, а значит, легче переносится в другие предметные области и на другие задачи. С другой стороны, чем более конкретную информацию использует метод, тем потенциально «точнее» он может выявить контекст. «Точнее» здесь употребляется с целью отразить тот факт, что при выявлении возможного значимого или допустимого незначимого контекстов для достижения наибольшего сокращения пространства поиска нужно стремиться минимизировать мощность первого и максимизировать мощность второго контекста. То есть «точнее» означает ближе к экстремуму.

К информации *о задачах планирования в целом* относятся знания о том, что всякая задача планирования включает в себя *описание* начального состояния и целевого условия. Эти описания можно формализовать различными способами, например, при помощи формул исчисления предикатов или на языке PDDL. Для простоты будем полагать, что описание начального состояния (как и всех других состояний) представляет собой множество «атомарных» фактов, выраженных кортежами вида: $\langle \mathbf{predicate}^{\mathbf{k}}\ \mathbf{arg}_1, \mathbf{arg}_2, \ldots, \mathbf{arg}_k \rangle$, где $\mathbf{arg}_i$ — имена объектов моделируемой предметной области ($1 \le \mathbf{i} \le \mathbf{k}$), а **predicate** — имя представляющего данный факт отношения между этими объектами. Как и в случае с моделью схемы действий верхний индекс у предиката означает арность описываемого им отношения. Аналогичным образом задается и целевое условие, с той лишь разницей, что кортежи помечаются модальностями «должен» или «не должен» факт присутствовать в целевом состоянии. Синтаксически, будем помечать модальность «не должен» значком отрицания «¬»; если значок отсутствует, то это значит, что имеет место

модальность «должен». Цель задачи планирования — достичь такого состояния, описание которого содержало бы *все* кортежи целевого условия с модальностью «должен» и не содержало ни одного кортежа целевого условия с модальностью «не должен».

Информация *о классах задач* может быть полезна в случае, если существует единый механизм выявления контекста для задач, относящихся к одному классу. Тогда получение контекста для текущей задачи будет сводиться к двум операциям: классификации задачи и получения контекста при помощи метода, созданного специально для данного класса задач.

Информация *о текущей решаемой задаче* включает описание начального состояния и цели текущей *конкретной* задачи планирования. Описание начального состояния в частности декларирует, какие объекты включены в модель, какими свойствами они обладают в начальном состоянии, и как они взаимосвязаны в начальном состоянии. Описание цели, соответственно, декларирует, какими свойствами должны обладать и какими отношениями должны быть связаны объекты в желаемом состоянии.

В данной работе рассматривает несколько методов выявления контекстов, которые можно классифицировать следующим образом.

1. Методы на базе закодированных эвристик.
2. Методы на базе эвристик, основанных на интерпретации набора правил, определяющих контекст. В этом случае знания о том, что должен включать в себя контекст, отделяются от формирующих контекст алгоритмов, что делает подход более универсальным. В качестве неформального примера правила такого рода можно привести фразу — «если в описании цели содержится атомарный факт, предикатом которого является *находиться_в*, то возможный значимый контекст должен содержать действия, порожденный от схемы *переместиться_в*».
3. Методы, опирающиеся на рассуждения по прецедентам.

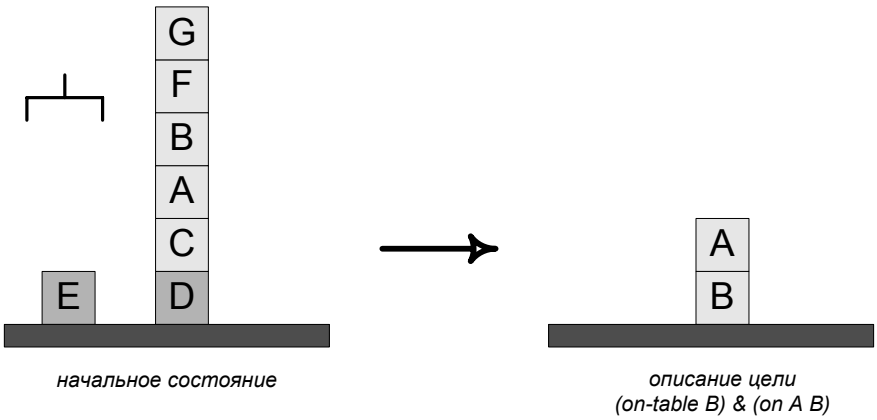
6.1 Методы на базе закодированных эвристик

Единственным преимуществом методов на базе закодированных эвристик является то, что они могут опираться на знания о предметной области, не представленные в постановке задачи планирования явным образом (по крайней мере, не представленные формально). Например, для домена с кубиками можно предложить довольно простую эвристику, выявляющую возможный значимый контекст. Обозначим **GoalObjs** — множество имен объектов, фигурирующих (в качестве аргументов) в атомарных фактах цели. Обозначим **InitObjs** — множество имен объектов, которые в начальном состоянии

находятся либо «непосредственно под» объектами из **GoalObjs**, либо «непосредственно или опосредованно над» ними. Наконец, обозначим **SignificantObjs** — объединение множеств **GoalObjs** и **InitObjs**. Тогда возможный значимый контекст составляют действия

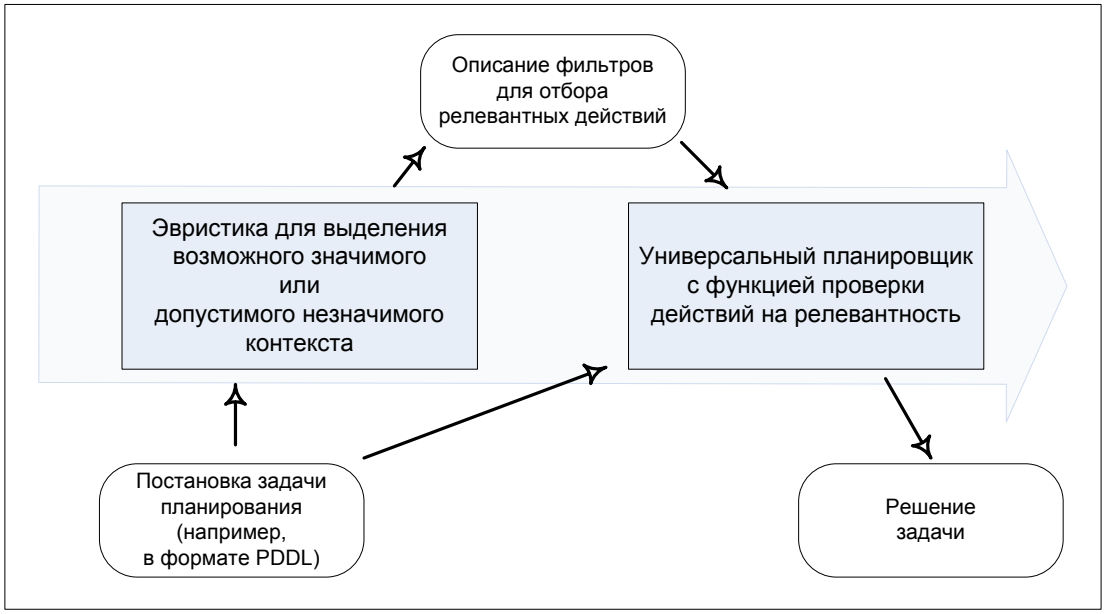
$$\mathbf{SCx} = \{\mathbf{a} \mid \mathbf{a} \in \mathbf{Actions},\ \ \mathbf{a} \in \bigcup_{obj \in SignificantObjs} \mathbf{actions}_{obj}\}$$

Эта эвристика оперирует такими понятиями как отношения «непосредственно над», «опосредованно над» и «непосредственно под». Формально в постановке задач для домена с кубиками такие отношения не присутствуют (хотя их можно определить через отношение «на»). Однако эти дополнительные знания могут быть закодированы в программном коде планировщика и, таким образом, использованы с целью выявления контекста.



Для данной задачи возможный значимый контекст (согласно предложенному выше описанию) состоит из всех действий, манипулирующих кубиками A, B, C, F и G. Действия с F и G необходимы, чтобы добраться до кубиков A и B, а из действий с кубиком C для решения задачи потребуется unstack(A,C). Действия по манипуляции с кубиками D и E при поиске решения можно не рассматривать; они составляют допустимый незначимый контекст.

Из-за того, что знания о принципах выявления контекста, закодированы в программном коде планировщика, такой метод недостаточно гибок. Тем не менее, этот подход может быть полезен при создании специализированных планировщиков в некоторых предметных областях. Общей методологией здесь может быть следующая последовательность действий. Сначала специальные «закодированные» эвристики выделят релевантные (или не релевантные) решаемой задаче действия. Затем найденный контекст может быть передан любому *универсальному* планировщику как описание спектра возможной (или наоборот, «запрещенной») деятельности.



6.2 Эвристические методы, основанные на интерпретации набора правил, определяющих контекст

Шагом в сторону более гибкого метода является создание эвристик, опирающихся на набор декларативных правил, определяющих связь между формальной постановкой задачи и контекстом. В этом случае знания, позволяющие выявить контекст, также являются ориентированными на одну предметную область. Однако декларативное представление этих знаний позволяет легко сменить предметную область, написав новые правила, но не изменяя алгоритмов. В отличие от закодированных эвристик, наличие единого механизма интерпретации правил (а значит, и наличие некоторого формального языка описания правил с определенными выразительными способностями) накладывает ограничение на множество эвристик, которые могут быть представлены в виде правил.

Что касается языка описания правил, то предварительно можно отметить следующее. Во-первых, такой язык может опираться на описания начального состояния и цели. Для того чтобы работать с этими описаниями, введем две новые функции. Пусть **P** — некоторая задача планирования. Тогда функция **InInit**(**P**) возвращает *описание начального состояния* в виде множества кортежей, описывающих атомарные факты. В свою очередь функция **InGoal**(**P**) возвращает *описание цели* в виде множества целевых условий, представленных аналогичными кортежами, которые (дополнительно) могут быть помечены модальностью «не должен».

Во-вторых, было бы полезно иметь возможность писать правила как ориентированные на один определенный класс задач, так и правила, осуществляющие классификацию задач (для одного класса задач контекст формируется по одним правилам,

для другого — по другим). Чтобы реализовать это, можно выбрать следующий общий вид для правил:

Если ⟨классификатор задачи⟩, ***то*** ⟨определитель контекста⟩.

Здесь ⟨классификатор задачи⟩ — это некоторое условие, позволяющее определить, принадлежит ли данная задача определенному классу. Если это условие определено как «*Истина*», то мы получим универсальное правило домена, которое срабатывает при решении задач всех классов.

В качестве примера работы классификатора можно привести следующую ситуацию. Пусть в некотором домене мы хотим различать задачи перемещения и ремонта. Обозначим «*at*» — имя отношения, отражающего местоположения объектов в домене, а «*broken*» — имя отношения, отражающее состояние исправности объекта. Тогда левые части правил определения контекста могли бы иметь следующий вид.

Если $\exists\, X,Y,Z.\ (\langle at\ X\ Y\rangle \in \mathbf{InInit}(\mathbf{P}) \wedge \langle at\ X\ Z\rangle \in \mathbf{InGoal}(\mathbf{P}) \wedge (Y \neq Z))$, ***то*** …

Если $\exists\, X.\ (\langle broken\ X\rangle \in \mathbf{InInit}(\mathbf{P}) \wedge \neg\langle broken\ X\rangle \in \mathbf{InGoal}(\mathbf{P}))$, ***то*** …

Здесь **P** — текущая (решаемая) задача планирования. Первое правило, фактически, декларирует следующее «если задача является перемещением, то…». В терминах постановки задачи планирования принадлежность классу «задача перемещения» определяется так. *Если существует некоторый объект X, который в начальном состоянии находился в месте Y, а согласно целевому условию должен находиться в другом месте Z, то мы имеем дело с задачей перемещения*. Второе правило аналогичным образом выполняет проверку принадлежности задачи к классу «ремонт». *Если существует некоторый объект X, который в начальном состоянии сломан, а должен стать исправным, то мы имеем дело с задачей ремонта*. Для первого класса задач возможный значимый контекст можно определить как, все действия, касающиеся перемещений, а для второго — ремонтов. Правые части предложенных правил должны содержать соответствующие конструкции для формирования такого рода контекстов.

Теперь рассмотрим вторую часть правила, которая называется ⟨определитель контекста⟩. Она должна определять, из чего состоит контекст в зависимости от конкретной решаемой задачи. Но здесь речь идет не о зависимости классифицирующего характера (как в левой части правила), а о зависимости от конкретного содержания предметной области. Так как в классическом подходе к планированию (и это отражено в синтаксисе языка PDDL) объекты предметной области, с которыми сталкивается планировщик, объявляются как часть *задачи планирования* (а не домена), то, решая различные задачи даже одного класса, мы можем иметь дело с различными объектами. Таким образом, ограничения на контекст вида $\mathbf{actions}_{\mathbf{Obj}}$ (и другие, в которых

фигурируют имена объектов) должны быть связаны с конкретной текущей задачей (и объявленными в ней объектами).

В связи с этим, элемент ⟨определитель контекста⟩ удобно разделить на две компоненты: ⟨декларатор объектов⟩ и ⟨модификатор контекста⟩. Синтаксически в правилах будем разделять их стрелкой:

⟨определитель контекста⟩ ::= [⟨декларатор объектов⟩ →] ⟨модификатор контекста⟩.

Элемент ⟨декларатор объектов⟩ является необязательным. Он определяет множества объектов, которыми будет пользоваться ⟨модификатор контекста⟩. Если ⟨модификатор контекста⟩ не содержит ограничений, в которых фигурируют объекты, то ⟨декларатор объектов⟩ опускается.

Рассмотрим на примерах, что может представлять из себя ⟨определитель контекста⟩. Пусть мы хотим определить допустимый незначимый контекст как «множество действий, которые манипулируют такими объектами, которые в начальном состоянии были красного цвета». Этот определитель можно записать при помощи конструкции вида:

$$\text{Set1}=\{X \mid \langle red\ X\rangle \in \mathbf{InInit(P)}\} \rightarrow \mathbf{NCx} = \bigcup_{\text{obj}\in\text{Set1}} \mathbf{actions}_{\mathbf{obj}}.$$

Здесь в левой части определителя (до стрелки) декларируется множество красных объектов (Set1), а в правой части объявляется, что допустимый незначимый контекст (**NCx**) состоит из действий, манипулирующих объектами из Set1.

Следующий пример демонстрирует ⟨определитель контекста⟩ без использования элемента ⟨декларатор объектов⟩. Пусть мы хотим определить возможный значимый контекст как «множество всех действий, порожденных от схемы действий *repair*». Такой определить может иметь следующий вид

SCx = **actions**[*repair*].

Язык обретет еще большую выразительную силу, если будет допускать инструкции модификации контекста (как множества). Это полезно, если одно правило определяет контекст, а другое его уточняет. С учетом этого ⟨модификатор контекста⟩ может принимать вид **SCx** = **SCx** ∪ **actions**[**schema**$^{\mathbf{k}}$] и тому подобные (с учетом разнообразия операций над множествами, видов контекстов и видов ограничительных множеств). Постулируем, что до применения набора правил множества **SCx** и **NCx** пусты.

Возвращаясь к аспектам языка описания правил, следует отметить, что для правил должен существовать порядок их применения. Это обусловлено тем, что контекст может

определяться путем постепенных уточнений. Такая возможность может быть реализована посредством нумерации правил.

Для каждого класса задач правила должны определять либо только возможный значимый контекст, либо допустимый незначимый.

В данной работе не приводится завершенной формализации языка описания правил и, возможно, в будущем обнаружатся возможности установления еще более глубоких и изощренных зависимостей между формальной постановкой задачи планирования и значимым (или незначимым) контекстом.

6.3 Методы, опирающиеся на рассуждения по прецедентам

Так как «рассуждения по прецедентам» (CBR – от case-based reasoning) являются методологией, можно предложить множество различных конкретных техник выявления контекстов на ее базе.

Например, CBR можно использовать только для классификации задач (то есть сопоставлять текущей задаче ее класс), но для каждого класса задач выявление контекста осуществлять при помощи набора декларативных правил. В этом случае придется использовать рассуждения по примерам (это частный случай рассуждений по прецедентам). То есть наполнение библиотеки прецедентов осуществляет эксперт (по предметной области), который должен внести достаточное количество примеров задач, сопоставив им их класс. В этом варианте фаза сохранения прецедентов отсутствует. Впрочем, для классификации можно использовать и другие методы.

Другим способом использования CBR может быть сопоставление задаче самого контекста, как множества. Но такой путь порождает сложную подзадачу автоматического получения значимого контекста по готовому решению задачи планирования на этапе сохранения прецедента. Можно, например, взять из решения все использованные действия, и объявить их возможным значимым контекстом. Можно также в качестве **SCx** объявить более обширный набор действий; например, все действия, которые оперируют встретившимися в решении объектами. Однако данный подход порождает следующую проблему. Полученные таким образом контексты будут оперировать конкретными объектами, а значит при решении другой задачи планирования с другими объектами, этими контекстами мы воспользоваться не сможем. Поэтому предлагаемый подход получения контекста из плана ограничен тем, что все задачи должны быть определены для *одного* набора объектов.

Возможным решением предыдущей проблемы мог бы быть отбор не самих объектов из решения, а декларация объектов, обладающих определенными характеристиками (в начальном состоянии). Например, если действия плана

манипулируют объектами *X* и *Y*, которые в начальном состоянии обладали характеристиками {*X*: зеленый, большой, железный} и {*Y*: синий, высокий}, то можно декларировать возможный значимый контекст, как множество действий, манипулирующих либо «зелеными, большими, железными» объектами, либо «синими и высокими». Такой путь также имеет проблему — как различить важные характеристики от незначительных (с точки зрения определения контекста).

Таким образом, в рамках методологии CBR сложно найти достаточно общие решения для задачи выявления контекста, что, однако, не исключает возможности построения на ее базе средств извлечения контекста для отдельных предметных областей и задач. По-видимому, такие средства должны опираться на эвристики на отдельных этапах CBR-цикла (при поиске подходящего прецедента(ов) и при сохранении прецедента).

7 Использование контекстов

Можно предложить два пути использования возможного значимого контекста; назовем их «константный» и «инкрементальный».

В первом случае предполагается, что возможный значимый контекст извлекается *точно*, то есть решение гарантированно существует. Тогда нам достаточно взять любой универсальный алгоритм планирования и отыскать решение в сокращенном (за счет удаления нерелевантных действий) пространстве поиска.

Второй случай допускает *неточное* выявление возможного значимого контекста (абстрактная задача может не иметь решения). Такое может происходить в случае использования CBR-методологии, а также эвристик или набора правил, не гарантирующих точное определение контекста. В этом случае мы можем в процессе поиска плана постепенно расширять возможный значимый контекст (вплоть до добавления всех исходных действий). Это дает нам полноту поиска (если, конечно, универсальный алгоритм планирования является полным), даже если мы сначала неточно определили контекст. Расширение контекста может осуществляться разными способами: на базе эвристик, обобщающих и иных правил или даже путем постепенного добавления действий или их групп, отобранных случайным образом.

Литература

[Bacchus, Yang, 1994] F. Bacchus and Q. Yang. Downward refinement and the efficiency of hierarchical problem solving. Artificial Intelligence, 71(1):43–100, 1994.

[Bonet, Geffner, 1998] B. Bonet, H. Geffner. HSP: Heuristic Search Planner. Entry at the AIPS-98 Planning Competition, Pittsburgh 6/98.

[Bylander, 1991] T. Bylander. Complexity Results for Planning. // In Proc. IJCAI'91, pp. 274-279, 1991.

[Bylander, 1994] T. Bylander. The Computational Complexity of Propositional STRIPS Planning. Artificial Intelligence 69(1-2):165-204, 1994.

[Choueiry el al., 2005] Berthe Y. Choueiry, Yumi Iwasaki, Sheila A. McIlraith: Towards a practical theory of reformulation for reasoning about physical systems. Artificial Intelligence, 162(1-2): 145-204, 2005.

[Erol et al., 1991] K. Erol, D. S. Nau, V. S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning: A detailed analysis. Tech. Rep. CS-TR-2797, UMIACS-TR-91-154, SRC-TR-91-96, Computer Science Department and Institute for Systems Research, University of Maryland, 1991.

[Fink, Yang, 1992] Eugene Fink and Qiang Yang. Automatically abstracting effects of operators. //In Proc. First International Conference on AI Planning Systems (AIPS-92), pp. 243–251, 1992.

[Giunchiglia et al., 1997] F. Giunchiglia, A. Villafiorita, T. Walsh. Theories of Abstraction. Technical Report MRG/DIST #97-0051, 1997.

[Giunchiglia, Walsh, 1992] F.Giunchiglia and T. Walsh. A Theory of Abstraction. Artificial Intelligence, 57(2-3): 323-390, 1992.

[Hoffmann, Nebel, 2001] J. Hoffmann, B. Nebel. The FF Planning System: Fast Plan Generation Through Heuristic Search. Journal of Artificial Intelligence Research, vol. 14, pp. 253 - 302, 2001.

[Holte, Choueiry, 2003] Robert C Holte and Berthe Y Chouiery, Abstraction and reformulation in artificial intelligence. Philosophical Transactions of the Royal Society B: Biological Sciences, 358(1435): 1197–1204, 2003.

[Holte et al., 1994] R.C. Holte, C. Drummond, M.B. Perez, R.M. Zimmer and A.J. MacDonald, Searching With Abstractions: A Unifying Framework and New High-Performance Algorithm // Proceedings of the Canadian Artificial Intelligence Conference 1994, pp. 263-270, 1994.

[Holte et al., 1996] Robert C. Holte, T. Mkadmi, Robert M. Zimmer, Alan J. MacDonald, Speeding up Problem Solving by Abstraction: A Graph Oriented Approach. Artificial Intelligence, 85(1-2): 321-361, 1996.

[Knoblock, 1991] Craig A. Knoblock. Automatically Generating Abstractions for Problem Solving. PhD thesis, School of Computer Science, Carnegie Mellon University, 1991.

[Knoblock, 1994] Knoblock, C., Automatic generation of abstraction for planning. Artificial Intelligence, vol. 68, pp. 243–302, 1994.

[Sacerdoti, 1974] Sacerdoti, E., Planning in a hierarchy of abstraction spaces. Artificial Intelligence, vol. 5, pp. 115–135, 1974.

[Yang, Tenenberg, 1990] Qiang Yang and Josh D. Tenenberg. Abtweak: Abstracting a nonlinear, least commitment planner. In Proceedings of the Eighth National Conference on Artificial Intelligence, Boston, MA, pp. 204-209, 1990.

[Zucker, 2003] Zucker, Jean-Daniel. A grounded theory of abstraction in artificial intelligence. Philosophical Transactions of the Royal Society B: Biological Sciences, 358(1435): 1293–1309, 2003.