Н. О. Гаранина, Е. А. Сидорова, Подход к верификации семейства мультиагентных систем разрешения конфликтов,
Модел. и анализ информ. систем, 2016, том 23, номер 6, 703–714

Моделирование и анализ информационных систем. Т. 23, №6 (2016), с. 703–714
Modeling and Analysis of Information Systems. Vol. 23, No 6 (2016), pp. 703–714

DOI: 10.18255/1818-1015-2016-6-703-714

УДК 004.052, 519.179.2

Подход к верификации семейства мультиагентных систем разрешения конфликтов

Гаранина Н. О.[1,2], Сидорова Е.А.[1]

получена 13 марта 2016

Аннотация. В данной работе мы описываем метод верификации для семейств распределенных систем, которые порождаются контекстно-зависимой сетевой грамматикой специального вида. Эта грамматика содержит специальные нетерминальные символы — квази-терминалы. Квази-терминалы однозначно соответствуют терминалам грамматики и могут задавать процессы, которые определяются слиянием базовых процессов системы, в то время как нетерминалы задают сети параллельных композиций этих процессов. Данный метод верификации основан на техниках верификации моделей и абстракции. Абстрактная репрезентативная модель семейства систем зависит от задающей их грамматики и верифицируемых свойств системы. Эта модель симулирует поведение заданных систем таким образом, что свойства, которые выполняются в репрезентативной модели, также выполняются и во всех заданных системах. Проверку свойств репрезентативной модели можно осуществлять с помощью метода проверки моделей. Свойства порождаемых систем специфицируются с помощью универсальной логики ветвящегося времени $\forall CTL$ с конечными детерминированными автоматами в качестве атомарных формул. Мы показываем, что предложенный метод верификации можно применять для проверки некоторых свойств мультиагентных систем разрешения конфликтов, в частности, систем разрешения неоднозначностей при пополнении онтологий. Также показано, что этот подход можно использовать для верификации вычислений на подрешетках, являющихся подграфами решеток вычислений, например, для вычисления четности числа работающих процессов.

Ключевые слова: проверка моделей, контекстно-зависимая сетевая грамматика, мультиагентные системы, абстракция

Для цитирования: Гаранина Н. О., Сидорова Е.А., "Подход к верификации семейства мультиагентных систем разрешения конфликтов", *Моделирование и анализ информационных систем*, **23**:6 (2016), 703–714.

Об авторах:
Гаранина Наталья Олеговна, orcid.org/0000-0001-9734-3808, канд. физ.-мат. наук,
Институт систем информатики им. А.П. Ершова СО РАН,
проспект Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: garanina@iis.nsk.su

Сидорова Елена Анатольевна, orcid.org/0000-0001-8731-3058, канд. физ.-мат. наук,
Институт систем информатики им. А.П. Ершова СО РАН,
проспект Лаврентьева, 6, г. Новосибирск, 630090 Россия, e-mail: lena@iis.nsk.su

Благодарности:
[1]Работа выполнена при финансовой поддержке РФФИ (проект №15-07-04144).
[2]Работа выполнена при финансовой поддержке СО РАН (интеграционный проект №15/10 «Математические и методологические аспекты интеллектуальных информационных систем»).

Введение

Мотивацией нашей работы является задача разрешения неоднозначности в рамках задачи пополнения онтологий из данных, представленных текстами на естественном языке. В работе [7] мы описали алгоритмы анализа текста, порождающие систему информационных агентов, соответствующих экземплярам заданной онтологии и входным данным. Однако особенности естественного языка порождают неоднозначности при пополнении онтологий, и эти агенты предназначены их разрешать. Для разрешения неоднозначностей мы предлагаем оценивать контекст агентов-экземпляров, т.е. насколько агент связан с другими агентами посредством информации, которой он владеет, и выбирать среди альтернатив агента, наиболее интегрированного в текст. Мы разработали обобщенный алгоритм разрешения конфликтов в мультиагентных системах [9], специализацией которого является мультиагентный алгоритм разрешения неоднозначностей [8], удаляющий наименее интегрированных агентов из системы.

В процессе работы алгоритма все агенты в параллельном режиме исполняют довольно сложные протоколы с периодической локальной синхронизацией. Следовательно, необходимо использовать формальные методы для доказательства корректности этого алгоритма. В качестве техники верификации мы выбрали метод проверки моделей. Мы верифицируем довольно специфическую мультиагентную систему разрешения конфликтов. Работы по мультиагентным системам обычно фокусируются на поведении агентов, методах коммуникации между агентами, знании и мнениях агентов о других агентах и окружении и т.п. [6, 13]. Работы, касающиеся процесса разрешения конфликтов, обычно рассматривают этот процесс в терминах поведения агента, зависящего от его внутреннего состояния, методов рассуждений и аргументации, однако динамика связей агентов не исследуется [11]. Есть статьи, связанные с динамикой взвешенных связей, но эти связи однородны, и их изменение не влияет на внутреннее состояние агента [5]. С другой стороны, работы по изучению социальных сетей, в которых агенты связаны типизированными связями, не учитывают их вес [1]. Насколько нам известно, не существует работ по верификации алгоритмов разрешения конфликтов исследуемого нами типа.

Техника проверки моделей широко используется для верификации распределенных и мультиагентных систем [3]. В нашем случае необходимо верифицировать не одну мультиагентную систему, а бесконечное семейство таких систем. Для верификации бесконечных семейств распределенных сетей в работе [2] был предложен особый метод проверки моделей. Этот метод основан на использовании контекстно-свободных сетевых грамматик, порождающих семейства распределенных систем, а также на абстрагировании с помощью конечных автоматов. Идея метода состоит в конструировании инварианта сети, основанного на заданной грамматике. Этот инвариант симулирует поведение всех систем семейства согласованно с абстрактными функциями, ассоциированными с проверяемыми свойствами, выраженными в логике ветвящегося времени $\forall CTL$. Благодаря согласованной симуляции, свойства, выполняющиеся для репрезентативного инварианта, также выполняются для всех систем семейства. Однако авторы [2] исследовали только контекстно-свободные грамматики, тогда как наша модель мультиагентной системы порождается контекстно-зависимой грамматикой специального вида. В данной статье мы определяем такую грамматику, расширив стандартное определение контекстно-свободной сетевой грамматики из [2] понятиями квази-терминалов и оператора слияния. Мы показываем, что метод верификации из [2] также может быть использован для семейств систем, заданных этой новой грамматикой, поскольку новая грамматика обладает свойствами, аналогичными свойствам исходной.

Оставшаяся часть статьи организована следующим образом. В следующем разделе 1 мы даем основные определения. Раздел 2 представляет результаты для нового оператора слияния, использующегося в нашей контекстно-зависимой грамматике с квази-терминалами. В разделе 3 описано использование предложенного метода верификации для системы разрешения конфликтов и вычислений на подрешетках. В заключении 4 обсуждаются направления дальнейших исследований.

1. Основные определения

Дадим необходимые определения из [2] в адаптированном виде. Изменения касаются оператора слияния и квази-терминалов сетевой грамматики.

Определение 1.
Помеченная система переходов (LTS) — это структура $M = (S, R, ACT, S_0)$, где
– S — множество состояний,
– $S_0 \subseteq S$ — множество начальных состояний,
– ACT — множество действий, и
– $R \subseteq S \times ACT \times S$ — тотальное отношение переходов такое, что для каждого $s \in S$ существует действие a и состояния s', для которого $(s, a, s') \in R$ (обозначается как $s \xrightarrow{a} s'$).

Пусть L_{ACT} — это класс систем LTS со множеством действий, являющимся подмножеством ACT, $L_{(S,ACT)}$ — это подмножество систем L_{ACT} со множеством состояний, являющимся подмножеством S. Пусть $ACT_1, ACT_2 \subseteq ACT$ и заданы две LTS $M_1 = (S_1, R_1, ACT_1, S_0^1)$ и $M_2 = (S_2, R_2, ACT_2, S_0^2)$ в классе L_{ACT}.

Определение 2.

– Функция $\|$: $L_{ACT} \times L_{ACT} \mapsto L_{ACT}$ называется *функцией композиции,* и параллельная композиция двух систем $M_1 \parallel M_2$ имеет вид $(S_1 \times S_2, R', ACT_1 \cup ACT_2, S_0^1 \times S_0^2)$.
– Функция $\cup : L_{ACT} \times L_{ACT} \mapsto L_{ACT}$ называется *функцией слияния,* и слияние двух систем $M_1 \cup M_2$ имеет вид $(S_1 \cup S_2, R', ACT_1 \cup ACT_2, S_0^1 \cup S_0^2)$.

Определение R' зависит от точной семантики функций композиции и слияния. Пусть S^i — это слово длины i в алфавите S.

Определение 3.
Для заданного множества состояний S и множества действий ACT всякое подмножество $\bigcup_{i=1}^{\infty} L_{(S^i,ACT)}$ называется *сетью* для пары (S, ACT).

Дадим определение *контекстно-зависимой сетевой грамматики с квази-терминалами (CSNQ-грамматика)* для описания сетей. Множество всех систем LTS, выводимых в сетевой грамматике, образует сеть, которая также является системой LTS. Пусть S – это множество состояний и ACT — множество действий. *CSNQ-грамматика* $G = (T, Qt, t, N, P, S)$ – это грамматика, где

– T — конечное множество терминалов, каждый из которых является LTS из класса $L_{(S,ACT)}$, эти LTS называют *базовыми процессами*,

– Qt — множество квази-терминалов, каждый из которых является LTS из класса $L_{(S,ACT)}$, их слияние определяет LTS,

– отображение $t : Qt \mapsto T$ ассоциирует квази-терминалы с терминалами,

– N — конечное множество нетерминалов, каждый нетерминал определяет сеть,

– P — множество правил вывода следующего вида:

– $A \longrightarrow B \parallel_k C$, где $A \in N$, и $B, C \in T \cup Qt \cup N$, и $\parallel_i$ — функция композиции;

– $\{V_1, ...V_n\} \longrightarrow t(V_1) \cup_i ... \cup_i t(V_n)$, где для $j \in [1..n]$ $V_j \in Qt$, и $\cup_i$ — функция слияния.

– $S \in N$ представляет сеть, порожденную грамматикой.

Заметим, что такие грамматики контекстно-свободны относительно функций композиции и контекстно-зависимы относительно функций слияния. Отметим также, что, поскольку множество квази-терминалов может быть бесконечно, то, вообще говоря, множество правил вывода, содержащих квази-терминалы, также может быть не ограничено. Неограниченные множества правил могут задаваться, например, регулярными выражениями. Однако очевидно, что такие грамматики порождают конечные сети процессов.

Для задания свойств моделей, составленных из конечного, но неопределенного числа систем LTS, определим конечный автомат над алфавитом S.

Определение 4.
$D = (Q, q_0, \delta, F)$ — *детерминированный автомат* над S, где

– Q — множество состояний автомата,

– $q_0 \in Q$ — начальное состояние,

– $\delta \subseteq Q \times S \times Q$ — отношение переходов,

– $F \subseteq Q$ — множество допустимых состояний, и

– $L(D) \subseteq S^*$ — множество слов, допускаемых автоматом D.

Мы используем конечные автоматы над S для спецификации атомарных свойств состояний. Пусть D — это автомат над S. Состояние s выполнимо в D $(s \models D) \Leftrightarrow s \in L(D)$. Язык спецификации — универсальная логика ветвящегося времени $\forall CTL$ [2] с конечными автоматами над S в качестве атомарных формул. Синтаксис $\forall CTL$ включает формулы, построенные из булевских констант, атомарных формул, связок $\neg$, $\vee$, $\wedge$, модальностей ветвящегося времени $\mathbf{AX}\varphi$, $\mathbf{AG}\varphi$, и $\varphi\mathbf{AU}\psi$ со стандартной семантикой.

Напомним определение абстрактной LTS из [2]. Для простоты, пусть язык спецификации содержит единственную атомарную формулу D. Для заданного автомата $D = (Q, q_0, \delta, F)$ и слова $w \in S^*$ функция, индуцированная w на Q, $f_w : Q \mapsto Q$, определяется как $f_w(q) = q' \Leftrightarrow q \xrightarrow{w} q'$. Заметим, что $w \in L(D) \Leftrightarrow f_w(q_0) \in F$. Два состояния s и s' *эквивалентны* $s \equiv s' \Leftrightarrow f_s = f'_s$. Функция f_s называется *абстракцией* s и обозначается как $h(s)$. Отношение $\models$ расширяется на абстрактные состояния: $h(s) \models D \Leftrightarrow f_s(q_0) \in F$. Поэтому $s \models D \Leftrightarrow h(s) \models D$.

Пусть F_D — это множество функций, соответствующих детерминированному автомату D. Функция абстракции h, расширенная на F_D, определяется как $h(f) = f$ для $f \in F_D$ и расширение функций h на $(S \cup F_D)$ — это $h((a_1, a_2, ..., a_n)) = h(a_1) \circ ... \circ h(a_n)$. Далее мы рассматриваем системы LTS в сети N на паре $(S \cup F_D, ACT)$.

Определение 5. *(абстрактной LTS)*
Для данной LTS $M = (S^i, R, ACT, S_0)$ в сети N, соответствующая абстрактная LTS определяется как $h(M) = (S^h, R^h, ACT, S_0^h)$, где
– $S^h = \{h(s) | s \in S^i\}$ — множество абстрактных состояний,
– $S_0^h = \{h(s) | s \in S_0\}$, и
– отношение R^h определено следующим образом. Для всех $h_1, h_2 \in S^h$, и $a \in ACT$: $(h_1, a, h_2) \in R^h \Leftrightarrow \exists s_1, s_2 [h_1 = h(s_1) \wedge h_2 = h(s_2) \wedge (s_1, a, s_2) \in R]$.

M' *симулирует* M (обозначается как $M \preceq M'$), если и только если существует предпорядок симуляции $E \subseteq S \times S'$ ($(s, s') \in E$ обозначается как $s \preceq s'$), удовлетворяющий следующим условиям: для каждого $s_0 \in S_0$ существует $s_0' \in S_0'$ такой что $s_0 \preceq s_0'$. Для каждого s, s', если $s \preceq s'$, то
– $h(s) = h(s')$, и
– для каждого s_1 такого, что $s \xrightarrow{a} s_1$, существует s_1' такой, что $s' \xrightarrow{a} s_1'$ и $s_1 \preceq s_1'$.

2. Проверка моделей с оператором слияния

Первые два предложения леммы доказаны в [2], последнее доказано ниже:

Лемма 1.
1. $M \preceq h(M)$, т.е., $h(M)$ симулирует M.
2. Если $M \preceq M'$, тогда $h(M) \preceq h(M')$.
3. $M \cup M' \preceq h(M) \cup h(M')$

Доказательство пункта 3 очевидно: $M \cup M' \preceq h(M \cup M')$ в силу пункта 1, и $h(M \cup M') = h(M) \cup h(M')$.$\blacksquare$

Следующая теорема о выполнимости свойств в системе LTS и в системе, ее симулирующей, была доказана в [2]. Она остается верной для нашего метода, поскольку не связана с новой грамматикой.

Теорема 1.
Пусть φ - формула $\forall CTL$ над атомарной формулой D. Пусть M и M' - две системы LTS, такие что $M \preceq M'$. Пусть $s \preceq s'$. Тогда $s' \models \varphi$ влечет $s \models \varphi$.

Определение 6.
Оператор композиции или слияния $\bullet \in \{\cup, \|\}$ называется *монотонным* относительно предпорядка симуляции $\preceq$, если и только если для заданных систем LTS, таких что $M_1 \preceq M_2$ и $M_1' \preceq M_2'$, верно $M_1 \bullet M_1' \preceq M_2 \bullet M_2'$. Сетевая грамматика G называется *монотонной*, если и только если все ее правила используют только монотонные операторы композиции и слияния.

Модифицируем определения и результаты для синхронных систем из [2] в соответствии с оператором слияния. Нашей моделью будет специальный вид систем LTS, *машина Мура* $M = (S, R, I, O, S_0)$, такая что множества входов I и выходов O не пересекаются. Кроме того, машины имеют специальное внутреннее действие, обозначаемое τ. Множество действий $ACT = \{\tau\} \cup 2^{I \cup O}$, где каждое не внутреннее действие — это множество входов и выходов. Переход $s \xrightarrow{a} t$ из состояния s в машине M при $a = i \cup o$, так что $i \subseteq I$ и $o \subseteq O$ возможен, только если среда предоставляет входы i и машина M вырабатывает выходы o.

Для оператора слияния множества входов и выходов сливающихся машин также не должны пересекаться. Пусть $I \cap O' = \emptyset$ и $O \cap I' = \emptyset$. Слияние машин M и M', $M'' = M \cup M'$ определено следующим образом:

– $S'' = S \cup S'$,
– $S''_0 = S_0 \cup S'_0$,
– $I'' = I \cup I'$ и $O'' = O \cup O'$, и
– $s'' \xrightarrow{a''} s''_1$ — переход в R'' $\Leftrightarrow$ выполнено следующее: $s'' \xrightarrow{a} s''_1$ — переход в R и $s'' \xrightarrow{a'} s''_1$ — переход в R' для некоторых a, a', таких что $a'' = a$ или $a'' = a'$.

Лемма 2.
Слияние $\cup$ монотонно относительно $\preceq$.

Доказательство. Пусть заданы $M = (S, R, I, O, S_0)$, $M_1 = (S_1, R_1, I_1, O_1, S_{1,0})$, $M' = (S', R', I', O', S'_0)$, $M'_1 = (S'_1, R'_1, I'_1, O'_1, S'_{1,0})$ — четыре машины Мура. Предположим, что $M \preceq M_1$ и $M' \preceq M'_1$. Пусть $E \subseteq S \times S_1$ и $E' \subseteq S' \times S'_1$ — соответствующие отношения симуляции. Докажем, что $M \cup M' \preceq M_1 \cup M'_1$.

Мы считаем, что $(s'', s''_1) \in E'' \Leftrightarrow (s'', s''_1) \in E$ или $(s'', s''_1) \in E'$. Покажем, что E'' имеет требуемое свойство. Из определения следует, что для заданного состояния $s_0 \in S_0 \cup S'_0$, существует $s_{0,1} \in S_{0,1} \cup S'_{1,0}$ такое, что $(s_0, s_{0,1}) \in E \cup E'$. Предположим $(s, s_1) \in E \cup E'$.
(1) По предположению верно, что $h(s) = h(s_1)$.
(2) Пусть $s \xrightarrow{a''} t$ — переход в $M \cup M'$. Это означает, что существует переход $s \xrightarrow{a} t$ в M или переход $s \xrightarrow{a'} t$ в M', такой что $a'' = a$ или $a'' = a'$. По определению существует $t_1 \in S_1 \cup S'_1$ такой, что $s_1 \xrightarrow{a} t_1$ или $s_1 \xrightarrow{a'} t_1$, где $(t, t_1) \in E$ или $(t, t_1) \in E'$. Следовательно, $s_1 \xrightarrow{a''} t_1$ и $(t, t_1) \in E''$, что завершает доказательство. ■

Понятие репрезентатива делает возможным построение симуляционного инварианта. Для заданной CSNQ-грамматики G мы ассоциируем каждый символ A грамматики с *репрезентативным процессом* $rep(A)$. Адаптируем определение свойства монотонности для множества репрезентативных процессов CSNQ-грамматики:

– для каждого терминала и квази-терминала A: $h(rep(A)) \succeq h(A)$, и
– для каждого правила $A \longrightarrow B \parallel C$: $h(rep(A)) \succeq h(h(rep(B)) \parallel h(rep(C)))$.

Дополним доказательство следующей теоремы из [2] для CSNQ-грамматик:

Теорема 2.
Пусть G — монотонная грамматика и найдены репрезентативы для символов G, удовлетворяющие свойству монотонности. Пусть A — символ грамматики G, и a — система LTS, полученная из A с по правилам G. Тогда $h(rep(A)) \succeq a$.

Доказательство. Докажем, что $h(rep(A)) \succeq h(a)$. Поскольку $h(a) \succeq a$, результат следует из транзитивности. Пусть $A \Rightarrow^k a$, т.е. a выводимо из A за k шагов. Индукция по k.

[$(k = 0)$] Доказано в [2].

[$(k = 1)$] Пусть A, B — квази-терминалы в правиле $A, B \longrightarrow t(A) \cup t(B)$ и при этом $a = t(A) \cup t(B)$. Тогда результат следует из свойства монотонности и леммы 1.

[$(k \geq 1)$] Доказано в [2]. ■

Метод верификации совпадает с методом в [2], поскольку верны аналогичные теоремы, необходимые для его обоснования. Пусть задана монотонная грамматика G и φ — формула $\forall CTL$ с атомарными формулами $D_1, ..., D_k$. Чтобы проверить, что каждая LTS, выводимая в грамматике G, удовлетворяет φ, необходимо выполнить следующие шаги:

1. Для каждого символа A в G выбрать репрезентативный процесс $rep(A)$ и построить абстрактную систему LTS $h(rep(A))$ в соответствии с формулами $D_1, ..., D_k$.
2. Проверить, что множество репрезентативов удовлетворяет свойству монотонности. Теорема 2 влечет, что для каждого a, выводимого в грамматике G, верно $h(rep(S)) \succeq a$.
3. Выполнить проверку моделей на $h(rep(S))$ для спецификации φ. По теореме 1, если $h(rep(S)) \models \varphi$, то для всех систем LTS M, выводимых в грамматике G, верно $M \models \varphi$.

Для поиска монотонных репрезентативов используется алгоритм из [2], с $\{t(A)\}$ в качестве начального репрезентативного множества каждого квази-терминала A.

3. Примеры

В данном разделе мы приводим определения грамматик, задающих мультиагентную систему разрешения конфликтов и подрешеток, задаем базовые процессы порождаемых систем и формулы $\forall CTL$, выражающие свойства этих систем. Отметим, что порождающая грамматика для систем разрешения конфликтов имеет конечные множества квази-терминалов и правил, в отличие от грамматики для подрешеток. Описание построения конечных детерминированных автоматов, являющихся базисом для абстрактных функций состояний порождаемых систем, и согласованных репрезентативов для символов грамматик выходит за рамки данной статьи.

3.1. Мультиагентная система разрешения конфликтов

Подробное описание мультиагентного алгоритма разрешения конфликтов при пополнении онтологии приведено в [9], а соответствующая ему специализированная система разрешения неоднозначностей описана в [8]. В этой статье мы кратко опишем коммуникационную структуру системы агентов, без рассмотрения действий агентов по обработке входящих сообщений.

Пусть задано множество *агентов-участников*. Некоторые из агентов находятся *в конфликте*, соответствующем некоторой неоднозначности. *Агент-мастер* конструирует конфликтно-свободное множество агентов, учитывая интегрированность конфликтных агентов в систему. Эта интегрированность оценивается посредством вычисления весов и конфликтных весов агентов. Конфликт разрешается через удаление либо изменение слабых агентов. Агент-мастер выполняет основной протокол, конструируя конфликтно-свободное множество, тогда как остальные агенты выполняют протоколы вычисления весов.

Каждый агент-участник соединен с мастером двусторонним каналом. Агенты-участники связаны друг с другом помеченными связями типов, соответствующих их реакции на конфликт: удаление (rem-тип) и обновление (upd-тип). Каждая помеченная связь ациклична. Обрабатывание каждой конфликтной реакции, индуцированной определенной связью, рассматривается как отдельный базовый процесс. Агент-участник может быть объединением таких процессов. Этот факт задает структуру грамматики, порождающей семейство наших мультиагентных систем для разного числа агентов, связанных друг с другом различным образом. Агенты соединены двусторонними каналами, соответствующими их помеченным связям. Такая структура мультиагентной сети порождается следующей контекстно-зависимой грамматикой с квази-терминалами $G = (T, Qt, t, N, P, S)$. Пусть задано множество связей $C = \{c_1, ..., c_n\}$ и каждая связь c_i^k имеет конфликтный тип $k \in \{rem, upd\}$. Тогда для грамматики G:

– терминалы $T = \{master\} \cup \bigcup_{i=1}^{n} \{root_i, inter_i, leaf_i\} \cup vrtxs^i$, при этом $vrtxs^i = \{vrtx \mid vrtx = inter_j$ или $vrtx = leaf_j, j \in [1..n]\}$ и $|vrtxs^i| = i$,

– квази-терминалы $Qt = \bigcup_{i=1}^{n} \{Inter_i, Leaf_i\}$,

– ассоциативное отображение $t : Qt \mapsto T$ определяется как $t(Inter_i) = inter_i$, и $t(Leaf_i) = leaf_i$ для каждого $i \in [1..n]$,

– нетерминалы $N = \{S\} \bigcup_{i=1}^{n} \{ROOT_i, SUB_i\}$;

– множество правил вывода P для каждого $i \in [1..n]$:

1. $S \longrightarrow master \parallel_m ROOT_1 \parallel_m \ldots \parallel_m ROOT_n$
2. $ROOT_i \longrightarrow (ROOT_i \parallel_{c_i^k} SUB_i) \bigvee (root_i \parallel_{c_i^k} SUB_i)$
3. $SUB_i \longrightarrow (SUB_i \parallel_{c_i^k} SUB_i) \bigvee (Inter_i \parallel_{c_i^k} SUB_i) \bigvee$
$(SUB_i \parallel_{c_i^k} Leaf_i) \bigvee (Inter_i \parallel_{c_i^k} Leaf_i) \bigvee$
$(inter_i \parallel_{c_i^k} SUB_i) \bigvee (SUB_i \parallel_{c_i^k} leaf_i) \bigvee$
$(inter_i \parallel_{c_i^k} Leaf_i) \bigvee (Inter_i \parallel_{c_i^k} leaf_i) \bigvee (inter_i \parallel_{c_i^k} leaf_i)$
4. $\{V_1, ..., V_m\} \longrightarrow t(V_1) \cup ... \cup t(V_m) = vrtx^m$, где для каждого $j \in [1..m]$ $V_j \in \{Inter_i, Leaf_i\}$, и если $V_j = Inter_i$, то для каждого $l \in [1..m]$ верно $V_l \neq Leaf_i$ $(i \in [1..n])$.

Неформально, правила грамматики позволяют построить ациклические деревья связей с произвольным числом потомков, а затем слить между собой их вершины. Параллельная композиция агентов-процессов синхронна. Протоколы вычислений весов обладают высокой степенью параллелизма, поэтому очень важно доказать их завершаемость и корректную синхронизацию. Выполнимость этих свойств необходима для корректного вычисления весов. Запуск этих вычислений может быть смоделирован пересылкой фишек.

Состояния каждого базового процесса определяются значениями следующих переменных состояний:

– $Name :\ int$ — имя процесса;

– $Channel$: множество $\{name :\ int;\ c\_type :\ bool;\ dir :\ bool;\ agn :\ int;\ rmvd : bool\}$, где $name$ — метка связи, c_type ее тип, dir — направление: потомок $(dir = 0)$ или предок $(dir = 1)$ с именем agn, и $rmvd$ — статус отсутствия;

– $Rmvd$: $bool$ — статус отсутствия;
– $Active$: $bool$ — статус активности;
– $WasActive$: $bool$ — статус предыдущей активности.

Входные и выходные каналы базового процесса соответствуют именам, типам и направлениям $Channel$. При синхронной композиции базовых процессов с различными именами соответствующие каналы с одинаковыми именами объединяются. При слиянии процессов с одним и тем же именем $Name$ множества каналов и множества $Channel$ объединяются. Процессы с разными именами не могут сливаться и процессы с одним и тем же именем $Name$ не могут участвовать в параллельной композиции. Переходы определяются пересылкой и получением фишек через каналы. Начальное состояние — $(i, Channel, 0, 0, 0)$, где $Channel$ — непустое множество каналов с $Channel.rmvd = 0$, число каналов с $dir = 1$ не больше 1 и число каналов с $dir = 0$ может быть равным 0.

Мы хотели бы проверить следующие свойства, выраженные с помощью $\forall CTL$. Для протокола параллельного вычисления весов:

– $\mathbf{AF}(\{wasActive\}^* \wedge \mathbf{AXAF}\{\neg Active\}^*)$
(каждый агент был активен, и после этого все вычисления завершаются).

Для протокола параллельного вычисления конфликтных весов:

– $\mathbf{AF}\{\neg Active\}^*$ (все вычисления завершаются);
– $\mathbf{AG}\{Not2Rmvd\}^*$ (Каналы и агенты не удаляются дважды).

3.2. Подрешетка

Подрешеткой размера $n \times m$ называется граф, в котором не более $n \times m$ вершин, сопоставленных элементам множества $V_G \subseteq \{(i,j) \mid 0 \leq i < n, 0 \leq j < m\}$ так, что вершины, помеченные парами (i,j) и (i',j'), являются смежными только в том случае, когда $i = i' \wedge j = j' + 1$, или $i = i' \wedge j = j' - 1$, или $i = i' + 1 \wedge j = j'$, или $i = i' - 1 \wedge j = j'$. Нетрудно видеть, что подрешетка размера $n \times m$ — это подграф графа решетки размера $n \times m$, возможно, несвязный.

Сетевая грамматика с квази-терминалами позволяет построить подрешетку посредством задания ее ячеек, каждая из которых является решеткой квази-терминалов 2×2, и последующим слиянием этих ячеек в подрешетку по определенным правилам. Мы выделяем самый левый нижний процесс в подрешетке, который может являться инициатором вычислений, и обозначаем его как ini. Грамматика $G = (T, Qt, t, N, P, S)$, задающая подрешетки SG произвольного размера, определяется следующим образом:

– терминалы $T = \{ini, p_{lb}, p_{rb}, p_{lt}, p_{rt}\}$, где ini — процесс-инициатор и p_* — базовые процессы, ассоциированные с вершинами ячеек с соответствующим направлением каналов,

– квази-терминалы $Qt = \cup_{i\geq 0, j\geq 0}\{l_ib_j, l_it_j, r_ib_j, r_it_j\}$ задают вершины ячеек,

– ассоциативное отображение $t : Qt \mapsto T$ определяется для всех $i \geq 0, j \geq 0$ как $t(l_ib_j) = p_{lb}$, $t(l_it_j) = p_{lt}$, $t(r_ib_j) = p_{rb}$, $t(r_it_j) = p_{rt}$,

– нетерминалы $N = \{S, ICELL, CELL, IBOT, ITOP, BOT, TOP\}$;

– множество правил вывода P. Мы считаем, что 1) функция $\|_0$ задает параллельную композицию подсетей, не связанных коммуникационными каналами; 2) функция $\|_v$ задает параллельную композицию двух подсетей, состоящих из двух процессов и попарно связанных двумя “вертикальными” каналами v; 3) функция $\|_g$ задает параллельную композицию двух процессов, связанных “горизонтальными” каналами g; 4) функция слияния $\cup$ объединяет множество каналов сливающихся процессов.

- Правила задания ячеек:
  1. $S \longrightarrow ICELL \|_0 CELL$;
  2. $ICELL \longrightarrow ITOP \|_v IBOT$;
  3. $ITOP \longrightarrow l_0t_0 \|_g r_0t_0$;
  4. $IBOT \longrightarrow ini \|_g r_0b_0$;
  5. $CELL \longrightarrow (CELL \|_0 CELL) \bigvee (TOP \|_v BOT)$;
  6. $TOP \longrightarrow \bigvee_{i\geq 0, j\geq 1}((l_it_j \|_g r_it_j) \vee (l_it_j \|_g p_{rt}) \vee (p_{lt} \|_g r_it_j)) \bigvee (p_{lt} \|_g p_{rt})$;
  7. $BOT \longrightarrow \bigvee_{i\geq 1, j\geq 0}((l_ib_j \|_g r_ib_j) \vee (l_ib_j \|_g p_{rb}) \vee (p_{lb} \|_g r_ib_j)) \bigvee (p_{lb} \|_g p_{rb})$.
- Правила склеивания вершин ячеек:
  1. $\{r_{i-1}t_{j-1}, l_it_{j-1}, r_{i-1}b_j, l_ib_j\} \longrightarrow p_{rt} \cup p_{lt} \cup p_{rb} \cup p_{lb}$;
  2. $\{r_{i-1}t_{j-1}, l_it_{j-1}, r_{i-1}b_j\} \longrightarrow p_{rt} \cup p_{lt} \cup p_{rb}$;
  3. $\{r_{i-1}t_{j-1}, l_it_{j-1}, l_ib_j\} \longrightarrow p_{rt} \cup p_{lt} \cup p_{lb}$;
  4. $\{r_{i-1}t_{j-1}, r_{i-1}b_j, l_ib_j\} \longrightarrow p_{rt} \cup p_{rb} \cup p_{lb}$;
  5. $\{l_it_{j-1}, r_{i-1}b_j, l_ib_j\} \longrightarrow p_{lt} \cup p_{rb} \cup p_{lb}$;
  6. $\{r_{i-1}b_j, l_ib_j\} \longrightarrow p_{rb} \cup p_{lb}$;
  7. $\{r_{i-1}t_j, l_it_j\} \longrightarrow p_{rt} \cup p_{lt}$;
  8. $\{l_it_{j-1}, l_ib_j\} \longrightarrow p_{lt} \cup p_{lb}$;
  9. $\{r_it_{j-1}, r_ib_j\} \longrightarrow p_{rt} \cup p_{rb}$.

Пусть необходимо подсчитать четность числа активных процессов подрешетки SG, достижимых из процесса-инициатора. Эта задача может решаться посредством простого распределенного эхо-алгоритма [12]. Пусть $Dir = \{down, up, left, right\}$ — множество направлений. Состояния каждого базового процесса определяются значениями следующих переменных состояний.

– $Channels$: *set of* Dir — множество каналов;
– Nb : Dir — имя соседа;
– Nbs : *set of* Dir — смежные процессы;
– $Active$: *bool* — статус активности;
– $Token$: *bool* — четность активности.

Входные и выходные каналы базового процесса соответствуют направлениям каналов $Channels$. При синхронной композиции базовых процессов соединяются каналы $down$ и up, $left$ и $right$. При слиянии процессов множества каналов и множества $Channels$ объединяются. Переходы определяются пересылкой и получением фишек через каналы. Начальное состояние — $(Ch_0, null, Nbs_0, a, 0)$, где Ch_0 и Nbs_0 — непустые множества каналов, и статус активности $a \in \{0, 1\}$. *Процесс-инициатор* ini посылает одновременно всем смежным процессам из $Nbs_{ini} = \{up, right\}$ фишки со своим значением активности $Active_{ini}$: $Token_{ini} := Active_{ini}$. При получении от *процесса-соседа* $Nb_p \in Dir$ фишки $Token_{Nb}$ процесс p, ранее ее не получавший, добавляет к ее значению свое значение активности $Token_p := Token_{Nb} \oplus Active_p$, отсылает фишки с этим значением далее каждому смежному процессу, кроме соседа: $q \in Nbs \setminus \{Nb\}$, и ожидает их возвращения. Получая фишку от смежного процесса q, процесс p учитывает ее значение $Token_p := Token_p \oplus Token_q$ и удаляет q из Nbs_p. Когда множество смежных процессов Nbs_p становится пустым, это означает, что все фишки вернулись, и процесс возвращает полученный результат $Token_p$ соседу Nb_p. При повторном получении фишки ($Nb_p \neq null$) процесс возвращает отправителю в качестве ее значения значение 0, не меняющее четности включенных процессов. Отметим, что в описании этого алгоритма имена процессов p и q употребляются для читаемости описания, при этом в реальном алгоритме каждому процессу для выполнения действий достаточно имен его каналов. Сумма значений фишек, вернувшихся к процессу-инициатору, будет значением четности числа достижимых активных процессов, что выражается следующим равенством (CV): $Token_{ini} \oplus \bigoplus_{p\in SG} Token_p = 0$. Корректность этого вычисления выражается с помощью следующей формулы $\forall CTL$: $\mathbf{AG}(End \to CorVal)$, где End — конечный автомат для строк состояний, в которых в первом состоянии строки $Nb_{ini} = \emptyset$, а остальные состояния произвольны, а $CorVal$ — автомат, допускающий строки состояний, удовлетворяющие равенству CV.

4. Заключение

В данной статье представлен метод верификации для семейств распределенных систем, заданных контекстно-зависимой грамматикой с квази-терминалами. Этот метод может быть использован для верификации мультиагентной системы разрешения неоднозначностей при пополнении онтологий. Свойства системы выражены формулами логики $\forall CTL$.

Мы планируем реализовать предложенный метод с использованием инструмента проверки моделей SPIN и дать формальное обоснование корректности мультиагентного алгоритма разрешения неоднозначностей. Однако некоторые свойства, касающиеся взаимодействия агентов, выражаются в рамках такого подхода довольно неэффективно. Это служит причиной для попыток использования более выразительных формализмов. Еще одним направлением дальнейших исследований является развитие предложенного метода для других типов контекстно-зависимых грамматик, в частности, для грамматики с квази-терминалами в комбинации с индексированной грамматикой [10] или грамматикой с управляемыми правилами [4], с помощью которых можно задавать произвольные точные решетки.

Список литературы / References

[1] Bergenti F., Franchi E., Poggi A., “Selected models for agent-based simulation of social networks”, *Proc. of 3rd Symposium on Social Networks and Multiagent Systems (SNAMAS 2011)*, 2011, 27–32.

[2] Clarke E. M., Grumberg O., Jha S., “Verifying Parameterized Networks”, *ACM Transactions on Programming Languages and Systems*, **19**:5 (1997), 726–750.

[3] Clarke E. M., Grumberg O., Peled D., *Model Checking*, MIT Press, 1999.

[4] Dassow J., “Grammars With Regulated Rewriting”, *Formal Languages and Applications*, Studies in Fuzziness and Soft Computing, **148**, 2004, 249–273.

[5] De Gennaro M. C., Jadbabaie A., “Decentralized Control of Connectivity for Multi-Agent Systems”, *Proc. of 45th IEEE Conference on Decision and Control*, 2006, 3628–3633.

[6] Fagin R., Halpern J. Y., Moses Y., Vardi M. Y., *Reasoning about Knowledge*, MIT Press, 1995.

[7] Garanina N. O., Sidorova E .A., “Ontology Population as Algebraic Information System Processing Based on Multi-agent Natural Language Text Analysis Algorithms”, *Programming and Computer Software*, **41**:3 (2015), 140–148.

[8] Garanina N., Sidorova E., “An Approach to Ambiguity Resolution for Ontology Population”, *Proc. of 24th International Workshop on Concurrency, Specification, and Programming*, (CS&P 2015) (Rzeszow, Poland, 2015, Sep. 28–30), **1**, 2015, 27–32.

[9] Garanina N. O., Sidorova E. A., Anokhin S. A., “Conflict Resolution in Multi-agent Systems with Typed Connections for Ontology Population”, *Perspectives of System Informatics*, Lecture Notes in Computer Science, **9609**, 2016, 116–129.

[10] Hopcroft J. E., Ullman J. D., *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.

[11] Huhns M. N., Stephens L. M., “Multiagent Systems and Societies of Agents”, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, MIT Press, 1999, 79–120.

[12] Tel G., *Introduction to Distributed Algorithms*, Cambridge University Press, 2000.

[13] Wooldridge M., *An Introduction to Multiagent Systems*, Willey&Sons Ltd, 2002.

Garanina N. O., Sidorova E.A., "An Approach to Verification of a Family of Multi-agent Systems for Conflict Resolution", *Modeling and Analysis of Information Systems*, **23**:6 (2016), 703–714.

DOI: 10.18255/1818-1015-2016-6-703-714

Abstract. In the paper, we describe a verification method for families of distributed systems generated by context-sensitive network grammar of a special kind. This grammar includes special non-terminal symbols, so called quasi-terminals, which uniquely correspond to grammar terminals. These quasi-terminals specify processes which are merging of base system processes, in contrast to simple nonterminals which specify networks of parallel compositions of the processes. The method is based on model checking technique and abstraction. An abstract representative model for a family of systems depends on their specification grammar and system properties to be verified. This model simulates the behaviour of the systems in such a way that the properties which hold for the representative model are satisfied for all these systems. The properties of the representative model can be verified by model checking method. The properties of a generated system are specified by universal branching time logic $\forall CTL$ with finite deterministic automata as atomic formulas. We show the use of this method for verification of some properties of a multiagent system for conflict resolution, in particular, for context-dependent disambiguation in ontology population. We also suggest that this approach should be used for verification of computations on sub-grids which are sub-graphs of computation grids. In particular, we consider the computation of parity of the active processes number in a sub-grid.

Keywords: model checking, context-sensitive network grammar, multi-agent systems, abstractions

About the authors:
Natalia O. Garanina, orcid.org/0000-0001-9734-3808, PhD,
A.P. Ershov Institute of Informatics Systems, SB RAS
6 Acad. Lavrentjev pr., Novosibirsk 630090, Russia, e-mail: garanina@iis.nsk.su

Elena A. Sidorova, orcid.org/0000-0001-8731-3058, PhD,
A.P. Ershov Institute of Informatics Systems, SB RAS
6 Acad. Lavrentjev pr., Novosibirsk 630090, Russia, e-mail: lena@iis.nsk.su

Acknowledgments:
The research has been supported by Russian Foundation for Basic Research (grant 15-07-04144) and Siberian Branch of Russian Academy of Science (Integration Grant n.15/10 “Mathematical and Methodological Aspects of Intellectual Information Systems”).