Домашняя работа по курсу "Распределенные алгоритмы"

Павел Кочетков, Илья Вашуров
521 группа

10 июня 2016 г.

Алгоритм Дейкстры-Шолтена

```
var state_p : (active, passive) init if p = p_0 then active else passive;
sc_p : integer init 0 ;
father_p : P init if p = p0 then p else udef ;


S_p: { state_p = active }
begin
    send <mes, p> ;
    sc_p := sc_p + 1 ;
end


R_p: {Message <mes, q> reached process p }
begin
    receive <mes, q> ;
    state_p := active ;
    if father_p = udef then
        father_p := q
    else
        send <sig, q> to q
end


I_p: { state_p = active }
begin
    state_p := passive ;
    if sc_p = 0 then (*remove p from T*)
    begin
        if father_p = p then
                        Announce
        else
            send <sig, father_p> to father_p ;
        father_p := udef
    end
```

```
end


A_p: {Signal <sig , p> reached p}
begin
    receive <sig , p>;
    sc_p := sc_p - 1 ;
    if sc_p = 0 and state_p = passive then
        begin
        if father_p = p then
                        Announce
        else
            send <sig , father_p> to father_p ;
        father_p := udef
    end
end
```

Для всякой конфигурации алгоритма определим два множества

$$\begin{aligned} V_T = \ & \{p : father_p \neq udef\} \\ \cup \ & \{< mes, p > \text{ на этапе пересылки}\} \\ \cup \ & \{< sig, p > \text{ на этапе пересылки}\} \end{aligned}$$

и

$$\begin{aligned} E_T = \ & (p, father_p) : \{father_p \neq udef\} \wedge father_p \neq p \\ \cup \ & \{(< mes, p >, p) :< mes, pi > \text{ на этапе пересылки}\} \\ \cup \ & \{(< sig, p >, p) :< sig, p > \text{ на этапе пересылки}\} \end{aligned}$$

Доказать, что предикат P является инвариантом алгоритма Дейкстры-Шолтена

$$\begin{aligned} P \equiv \ & state_p = active \Longrightarrow p \in V_T && (1) \\ \wedge \ & (u, v) \in E_T \Longrightarrow u \in V_T \wedge v \in T \cap P && (2) \\ \wedge \ & sc_p = \#\{v : (v, p) \in E_T\} && (3) \\ \wedge \ & V_T \neq \emptyset \Longrightarrow T - \text{дерево с корнем } p_0 && (4) \\ \wedge \ & (state_p = passive \cap sc_p = 0) \Longrightarrow p \notin V_T && (5) \end{aligned}$$

Доказательство:

Начальное состояние:

(1): $\forall p \backslash \{p_0\} : state_p = passive$. $state_{p_0} = active$, при этом $father_{p_0} = p_0 \neq udef \rightarrow$ соотношение (1) выполняется

(2): $E_T = \emptyset$, поэтому (2) выполняется

(3): $\forall p : sc_p = 0$, поэтому (3) выполняется

(4): $V_T = \{p_0\}$ и $E_T = \emptyset$, поэтому (4) выполняется

(5): $state_{p_0} = active$, значит условие (5) выполняется (ложная предпосылка).

S_p:

(1): Выполнение S_p не влияет на (1) (ни один процесс не становится активным, из V_T не происходит удаление процессов), поэтому (1) выполняется

(2): В дереве появляется новая вершина <mes, p> и новая дуга (<mes, p>, p), ведущая к процессу, а значит верно (2)

(3): В V_T добавляется новая вершина <mes, p>. Так как количество вершин-последователей у вершины p увеличивается на 1, то и значение переменной sc_p должно увеличиться на 1, что и происходит в S_p. Таким образом, соотношение (3) выполняется

(4): S_p не удаляет вершин из V_T, а поэтому (4) остаётся верным

(5): Ни один процесс не меняет своего состояния и не добавляется в V_T, поэтому остаётся верным (5)

R_p:

(1): Состояние процесса $state_p = active$. Если $father_p \neq udef$, то процесс уже находился в V_T. В противном случае, процесс добавляется в V_T, поэтому (1) выполняется

(2): Если $father_p = udef$, то <mes, q> замещает процесс p, а процесс q становится его родительской вершиной, в противном же случае его замещает сообщение <sig, q>, для которого родительской вершиной также является q, а так как q $\in V_T$, то (2) верно

(3): Число вершин-потомков у процесса q не изменяется. В результате R_p вершина-последователь <mes, q> замещается либо процессом p, либо сообщением <sig, q>, поэтому (3) выполняется

(4): При замещении вершин, структура графа не изменяется, и поэтому T - по-прежнему дерево с корнем p_0 и (4) остаётся верным

(5): После выполнения R_p состояние $state_p = active$, а процесс либо уже был включен в V_T, либо включается в V_T. Таким образом, (5) выполняется.

I_p:

(1): Состояние процесса p становится $state_p = passive$, поэтому (1) выполняется

(2): Если $sc_p \neq 0$, выполнение R_p не влияет на (2). Если же $sc_p = 0$, то листовая вершина p, не являющаяся корнем, замещается сигналом <sig, $father_p$> от которого идёт дуга к процессу $father_p$, который по-прежнему содержится в V_T. Следовательно, (2) выполняется

(3): Если $sc_p \neq 0$, выполнение R_p не влияет на (3). Если же $sc_p = 0$ и вершина p не корневая, то она удаляется из T, но ее место занимает сигнал <sig, $father_p$>. Следовательно sc_p для родительской вершины сохраняет корректное значение. Если же вершина p корневая, то при её удалении сохраняется корректное значение $sc_p = 0$, а значит (3) верно

(4): Если $sc_p \neq 0$, выполнение R_p не влияет на (4). Если же $sc_p = 0$ и вершина не является корневой, то при её замещении на вершину <sig, $father_p$>, структура графа не изменяется. Если же вершина является корневой, то после выполнения действия I_p, вершина-корень будет удалена, и тогда $V_T = \emptyset$, а значит (4) верно

(5): Если $sc_p \neq 0$, то предпосылка (5) ложная. Если же $sc_p = 0$, то вершина удаляется из V_T, соответственно, (5) выполняется

A_p:

(1): A_p не изменяет состояния процесса p, и, если $state_p = active$, не производит удаления процесса из V_T. Следовательно, (1) выполняется

(2): A_p удаляет полученный сигнал из V_T с соответствующей дугой в E_T, а если $sc_p = 0$ и $state_p = passive$, то листовая вершина p, не являющаяся корнем, замещается сигналом <sig, $father_p$> от которого идёт дуга к вершине-родителю, которая по-прежнему содержится в V_T, а значит (2) верно

(3): При получении сигнала <sig, p>, количество вершин-потомков уменьшается на единицу, что соответсвует пересчету sc_p в A_p. Если же $sc_p = 0$ и $state_p = passive$ и вершина p не корневая, то вершина p удаляется из T, но ее место занимает вершина <sig, $father_p$>. Следовательно sc_p для родительской вершины сохраняет корректное значение и выпоняется (3)

(4): A_p удаляет вершину-последователь, соответстующую сигналу <sig, p>. Так как она является листом, то T остаётся деревом с корнем в p_0. Если $sc_p = 0$ и $state_p = passive$, то, если вершина p не корневая, происходит замещение вершины p на <sig, $father_p$> и структура графа при этом не изменяется, а если вершина является корнем, то после выполнения действия A_p, она будет удалена, и тогда $V_T = \emptyset$, поэтому (4) остаётся верным

(5): Если $sc_p = 0$ и $state_p = passive$, то вершина p удаляется из V_T, а значит (5) выполняется