Распределённые алгоритмы

mk.cs.msu.ru → Лекционные курсы → Распределённые алгоритмы

Блок 9

Как обосновывать корректность
распределённых алгоритмов

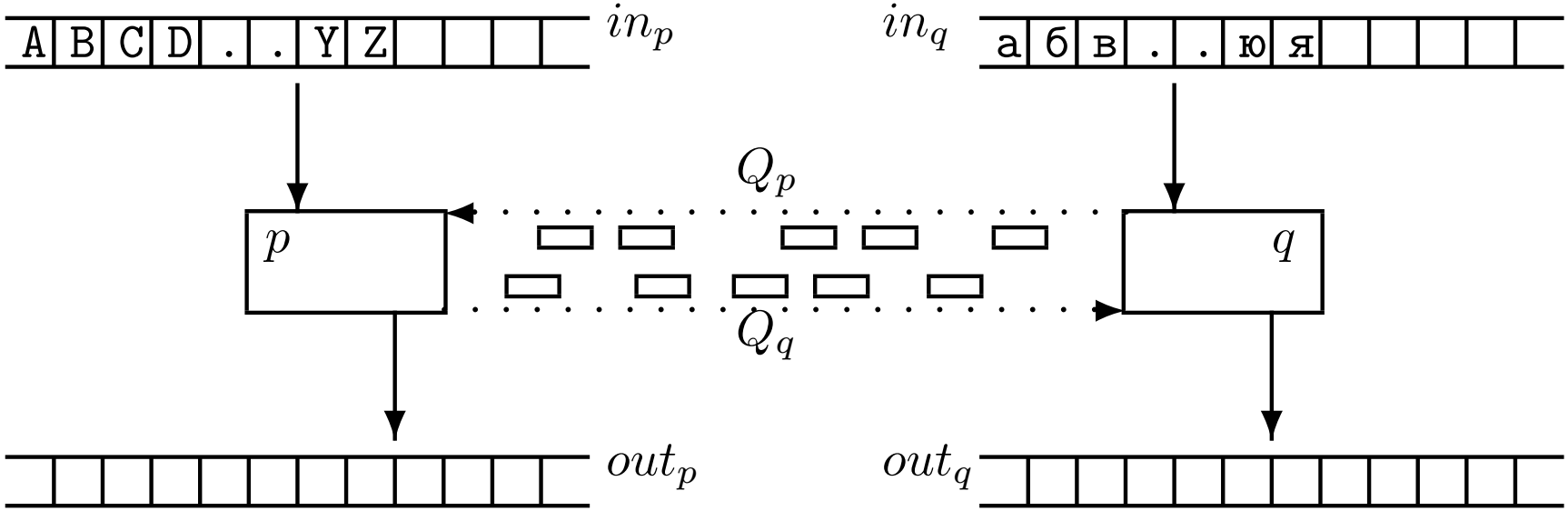
Свойства безопасности и живости

Свойства корректности
симметричного протокола раздвижного окна

Лектор:
Подымов Владислав Васильевич
E-mail:
valdus@yandex.ru

ВМК МГУ, 2023/2024, весенний семестр

Напоминание: симметричный протокол раздвижного окна



Узлам p и q требуется передать друг другу *потоки* данных in_p, in_q, записав их в массивы out_q и out_p соответственно

Как показать, что распределённый протокол раздвижного окна правильно решает поставленную задачу?

Можно поставить вопрос шире:

А как в целом можно доказывать корректность распределённых алгоритмов, если с.п., отвечающие р.а., слишком велики и непонятны для «прямого» анализа?

Свойства безопасности и живости

Даже если с.п. невозможно проанализировать явно «в лоб» (*как, например, для BSWP*), можно

- сформулировать свойства правильности этой с.п. и
- доказать, что с.п. удовлетворяет этим свойствам

Свойством тех или иных объектов (например, конфигураций или с.п.) будем называть произвольное множество P таких объектов, имея в виду, что объект x обладает свойством P, если $x \in P$

Соотношение $x \in P$ для свойства P будем также записывать в виде $P(x)$

Свойства объектов будем естественным образом представлять в виде утверждений, разбивающих объекты на обладающие этим свойством (те, для которых утверждение верно), и не обладающие (для которых утверждение неверно)

Свойства безопасности и живости

Для лаконичной записи утверждений иногда будем использовать элементы языка логики предикатов:

- $(P_1 \,\&\, P_2)$: верны оба утверждения P_1 и P_2
- $(P_1 \vee P_2)$: верно хотя бы одно из утверждений P_1, P_2
- $(\neg P)$: утверждение P неверно
- $(P_1 \Rightarrow P_2)$: если верно утверждение P_1, то обязательно верно и P_2
- $(\forall x \in X : P)$: для любого элемента x множества X верно утверждение P
- $(\exists x \in X : P)$: существует элемент x множества X, для которого верно утверждение P
- $(\forall x \in X, Q_1, \ldots, Q_n : P) = (\forall x \in X : (Q_1 \,\&\, \ldots \,\&\, Q_n \Rightarrow P))$
- $(\exists x \in X, Q_1, \ldots, Q_n : P) = (\exists x \in X : (Q_1 \,\&\, \ldots \,\&\, Q_n \,\&\, P))$

Приоритеты операций по убыванию: $\neg$; $\&$; $\vee$; $\Rightarrow$; $\forall$ и $\exists$

Свойства безопасности и живости

$\mathfrak{R}(S)$ — так будем обозначать множество всех достижимых конфигураций с.п. S

$\Pi(S)$ — так будем обозначать множество всех вычислений с.п. S

Свойство систем переходов S и соответствующих распределённых алгоритмов называется

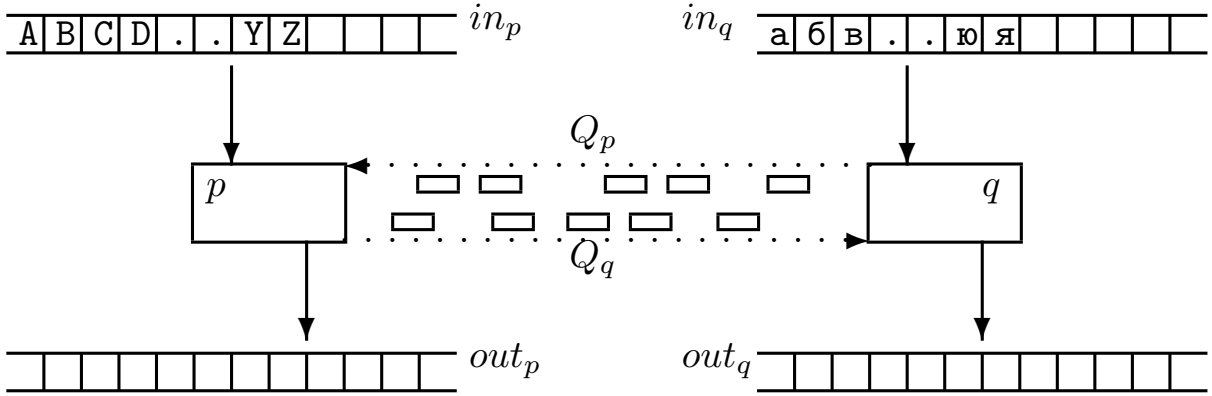
- свойством безопасности, если оно представлено в виде утверждения о том, что все достижимые конфигурации с.п. обладают заданным свойством P:
$$\mathfrak{R}(S) \subseteq P$$
- свойством живости, если оно представлено в виде утверждения о том, что в каждом вычислении с.п. хотя бы одна конфигурация обладает заданным свойством P:
$$\forall \pi \in \Pi(S) : \exists \gamma \in \pi : P(\gamma)$$

Обозначенное выше множество конфигураций P будем называть целевым множеством свойства живости или безопасности

Свойства безопасности и живости

Свойства правильности распределённых алгоритмов на практике нетрудно разбиваются на системы свойств безопасности и живости

Пример



Корректность BSWP можно выразить следующими свойствами

Безопасность BSWP: в каждой достижимой конфигурации для каждого номера i верно $out_p[i] \in \{in_q[i], \perp\}$ и $out_q[i] \in \{in_p[i], \perp\}$

$$\forall x \in \mathfrak{R}(S) : \forall i \in \mathbb{N}_0 : out_p[i] \in \{in_q[i], \perp\} \,\&\, out_q[i] \in \{in_p[i], \perp\}$$

Живость BSWP(k) для каждого номера k блоков данных, $k \in \mathbb{N}_0$: в любом вычислении содержится конфигурация, в которой все значения $out_p[0], \ldots, out_p[k], out_q[0], \ldots, out_q[k]$ отличны от $\perp$

$$\forall \pi \in \Pi(S) : \exists \sigma \in \pi : \forall i \in \{0, 1, \ldots, k\} : out_p[i] \neq \perp \,\&\, out_q \neq \perp$$

Обоснование свойств безопасности

Для множеств конфигураций P и Q с.п. $S = (\mathcal{C}, \mathcal{I}, \to)$ записью $P \to Q$ обозначим тот факт, что любой переход, исходящий из конфигурации, обладающей свойством P, обязательно ведёт в конфигурацию, обладающую свойством Q:

$$\forall (\gamma, \delta) \in \to : P(\gamma) \Rightarrow Q(\delta)$$

Множество конфигураций P будем называть монотонным относительно с.п. S, если верно утверждение $P \to P$

Множество конфигураций P с.п. S будем называть инвариантом с.п. S, если верно следующее:

1. $\mathcal{I} \subseteq P$
   - Все начальные конфигурации обладают свойством P
2. Множество P монотонно

Обоснование того, что с.п. обладает свойством безопасности, зачастую сводится к тому, чтобы предложить (с обоснованием) подходящий инвариант

Обоснование свойств безопасности

Теорема (о безопасности инварианта)
Если P — инвариант с.п. S, то $\mathfrak{R}(S) \subseteq P$

Доказательство

Предположим от противного, что P — инвариант, но $\mathfrak{R}(S) \not\subseteq P$

Тогда $\exists \gamma \in \mathfrak{R}(S) : \neg P(\gamma)$

Так как $\gamma \in \mathfrak{R}(S)$, то в S существует начальный путь $\gamma_1 \to \cdots \to \gamma_n$, такой что $\gamma_n = \gamma$

По определению инварианта, для всех i, $1 \leq i \leq n$, верно $P(\gamma_i)$

Значит, в частности, верно $P(\gamma_n)$

Но $\gamma_n = \gamma$, а значит, верно $\neg P(\gamma_n)$ (*противоречие*) ▼

Таким образом, если целевое множество свойства безопасности оказалось инвариантом с.п., то с.п. обязательно обладает этим свойством

Обоснование свойств безопасности

Теорема (о проверке безопасности с.п.)
Если P — инвариант с.п. S и Q — свойство конфигураций S, такое что $P \subseteq Q$, то верно $\mathfrak{R}(S) \subseteq Q$

Доказательство

Согласно безопасности инварианта, верно $\mathfrak{R}(S) \subseteq P$

Значит, если $P \subseteq Q$, то верно $\mathfrak{R}(S) \subseteq P \subseteq Q$ и, следовательно, $\mathfrak{R}(S) \subseteq Q$ ▼

Таким образом, чтобы обосновать, что с.п. $S = (\mathcal{C}, \mathcal{I}, \rightarrow)$ обладает свойством безопасности с целевым множеством Q, достаточно найти инвариант P, такой что $P \subseteq Q$

В терминологии, применяющейся к утверждениям, $P \subseteq Q$ означает, что из утверждения P, сформулированного относительно заданной конфигурации γ, следует утверждение Q для той же γ:

$$P \subseteq Q \quad \Leftrightarrow \quad \forall \gamma \in \mathcal{C} : P \Rightarrow Q$$

Обоснование свойств безопасности

Д.з. 1. Верно ли, что любое свойство, которым обладают все достижимые конфигурации с.п., является инвариантом? (Ответ обосновать.)

Д.з. 2. Докажите, что для любых инвариантов P_1, P_2 свойства конфигураций, представляющиеся утверждениями $(P_1 \vee P_2)$ и $(P_1 \,\&\, P_2)$, также являются инвариантами

Обоснование свойств живости

$$(\forall \pi \in \Pi(S) : \exists \gamma \in \pi : P(\gamma))$$

$\mathfrak{T}(S)$ — так будем обозначать множество всех тупиковых конфигураций с.п. S

Будем говорить, что с.п. S правильно завершает вычисления относительно свойства конфигураций P, если $\mathfrak{T}(S) \subseteq P$

Для обоснования живости конечных вычислений π с.п. S достаточно убедиться, что S правильно завершает вычисления относительно целевого множества P

- (чтобы доказать, что в вычислении содержится конфигурация из P, достаточно показать, что последняя конфигурация вычисления входит в P)

Если, кроме того, целевое множество P монотонно, то это достаточное условие оказывается и необходимым для конечных вычислений

А для проверки живости бесконечных вычислений с.п. придётся воспользоваться другим приёмом

Обоснование свойств живости

Частично упорядоченное множество (ЧУМ) $(W, <)$ называется фундированным (или, по-другому, обладающим свойством обрыва убывающих цепей), если не существует бесконечной последовательности, убывающей относительно $<$

$$(w_1 > w_2 > w_3 > \ldots;\ a > b \Leftrightarrow b < a)$$

Примеры фундированных множеств:

- $(\mathbb{N}_0, <)$ с естественным отношением $<$ сравнения чисел
- $(2^X, \subset)$, где X — конечное множество и 2^X — множество всех подмножеств X
- $(\mathbb{N}_0^n, <)$, где $n \in \mathbb{N}$ и $<$ — (линейное) отношение лексикографического порядка или (нелинейное) отношение покомпонентного сравнения наборов

Примеры нефундированных множеств:

- $(\mathbb{Z}, <)$
- $(\mathbb{N}_0, >)$
- $(2^{\mathbb{N}_0}, \subset)$

Обоснование свойств живости

Пусть заданы с.п. $S = (\mathcal{C}, \mathcal{I}, \to)$, целевое множество P и фундированное ЧУМ $(W, <)$

Тогда отображение $f : \mathcal{C} \to W$ называется функцией нормировки (относительно S, P и $(W, <)$), если для каждого перехода $\gamma \to \delta$ верно хотя бы одно из двух:

- $f(\gamma) > f(\delta)$
- $P(\delta)$

Иными словами, функция нормировки сопоставляет конфигурациям характеристику из W, строго убывающую в вычислениях системы до тех пор, пока не станет верно P

Обоснование свойств живости

Теорема (о проверке живости с.п.). Для любой с.п. S и любого целевого множества конфигураций P верно следующее: если S правильно завершает вычисления и для S существуют фундированное ЧУМ и функция нормировки, то верно утверждение $\forall \pi \in \Pi(S) : \exists \gamma \in \pi : P(\gamma)$

Доказательство.

Предположим от противного, что S правильно завершает вычисления и существуют фундированное ЧУМ $(W, <)$ и функция нормировки f, но при этом неверно $\forall \pi \in \Pi(S) : \exists \gamma \in \pi : P(\gamma)$

Тогда существует вычисление $\pi = (\gamma_1 \to \gamma_2 \to \dots)$ с.п. S, все конфигурации γ_i которого не обладают свойством P

Так как S правильно завершает вычисления, то последовательность π обязана быть бесконечной

Тогда $(f(\gamma_1), f(\gamma_2), \dots)$ — бесконечная убывающая последовательность фундированного ЧУМ $(W, <)$, чего быть не может по определению фундированности (*противоречие*) ▼

Обоснование свойств живости

Последней теоремой задаётся популярный способ обоснования живости распределённых алгоритмов и соответствующих систем переходов

Для обоснования живости достаточно сделать следующее:

1. Убедиться (обоснованно), что все тупиковые конфигурации обладают целевым свойством
2. Предложить (с обоснованием) какую-либо функцию нормировки