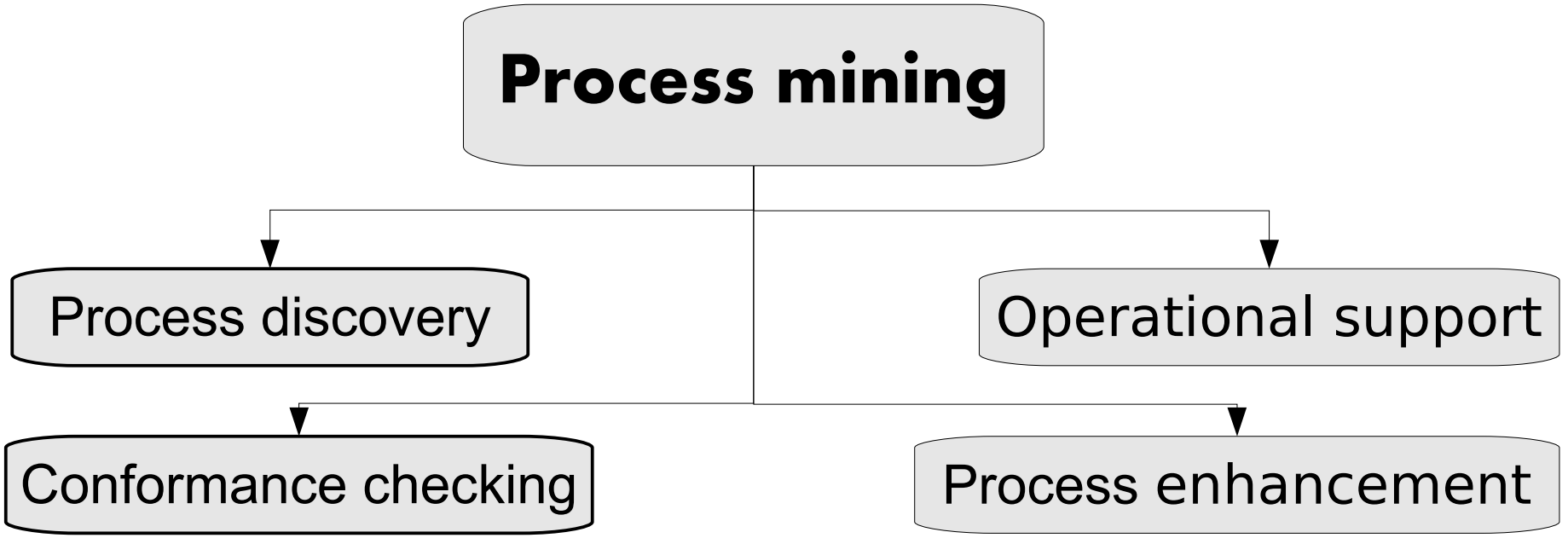
Распределенные процедуры извлечения и анализа процессов

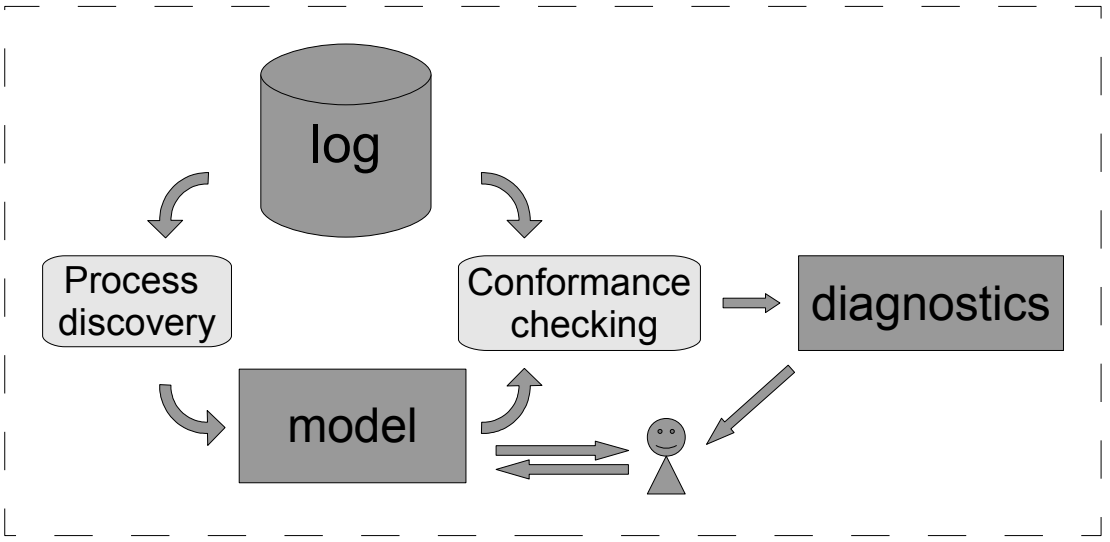
Лаборатория процессно-ориентированных информационных систем

Алексей Мицюк

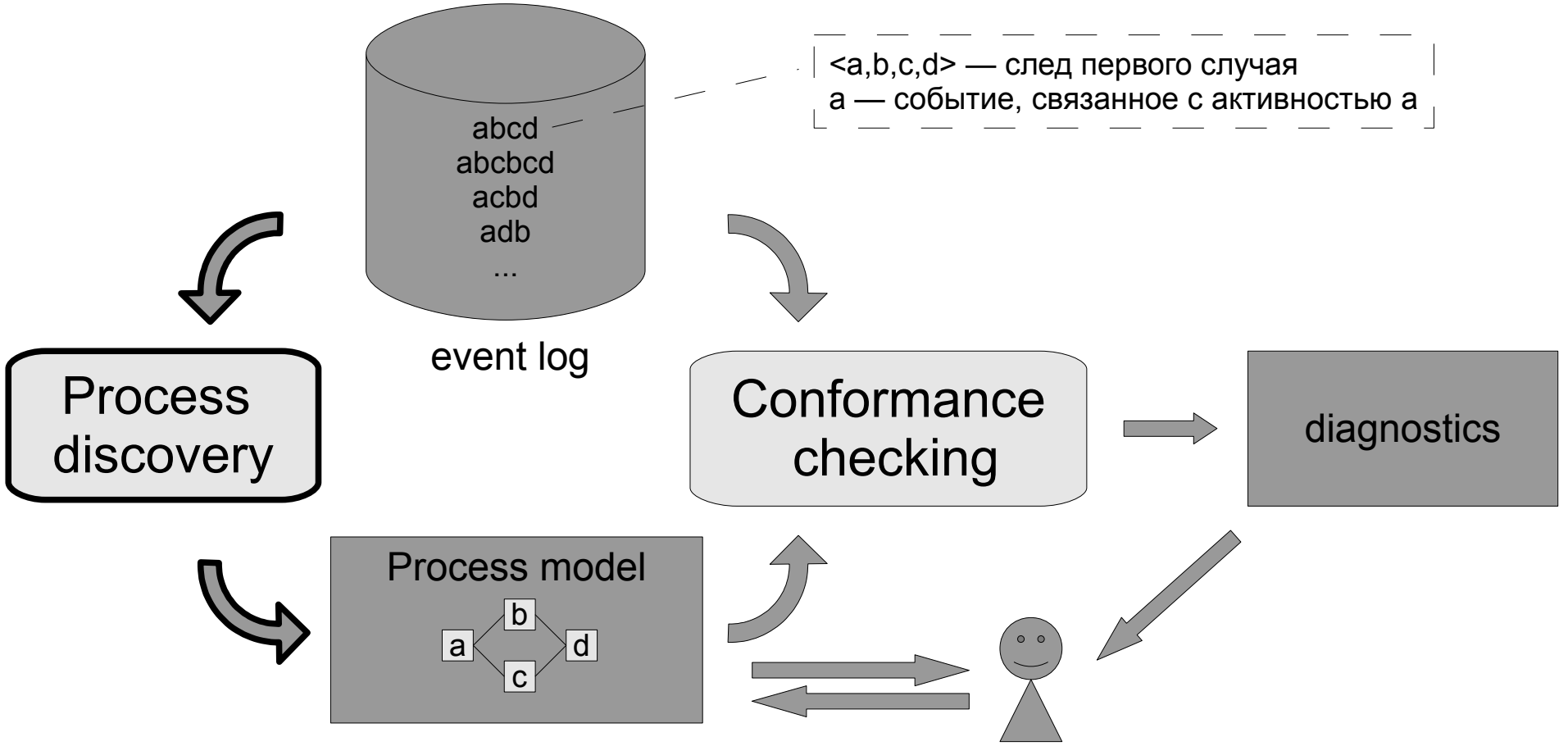
Москва, 2013

Основные разделы process mining



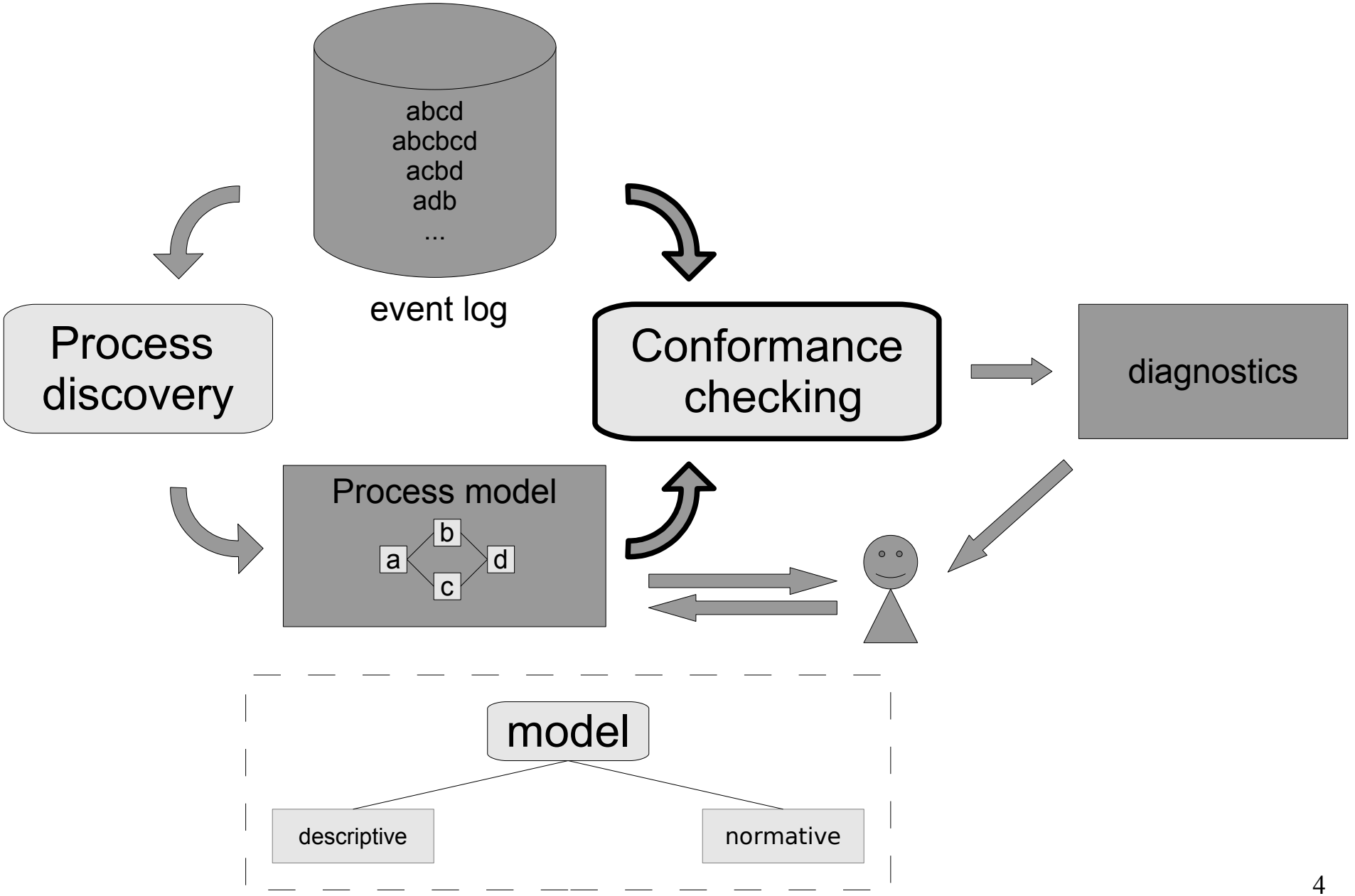


Процедуры извлечения и анализа процессов



fitness	simplicity	generalization	precision
соответствие логу событий	бритва Оккама	общность модели	точность модели

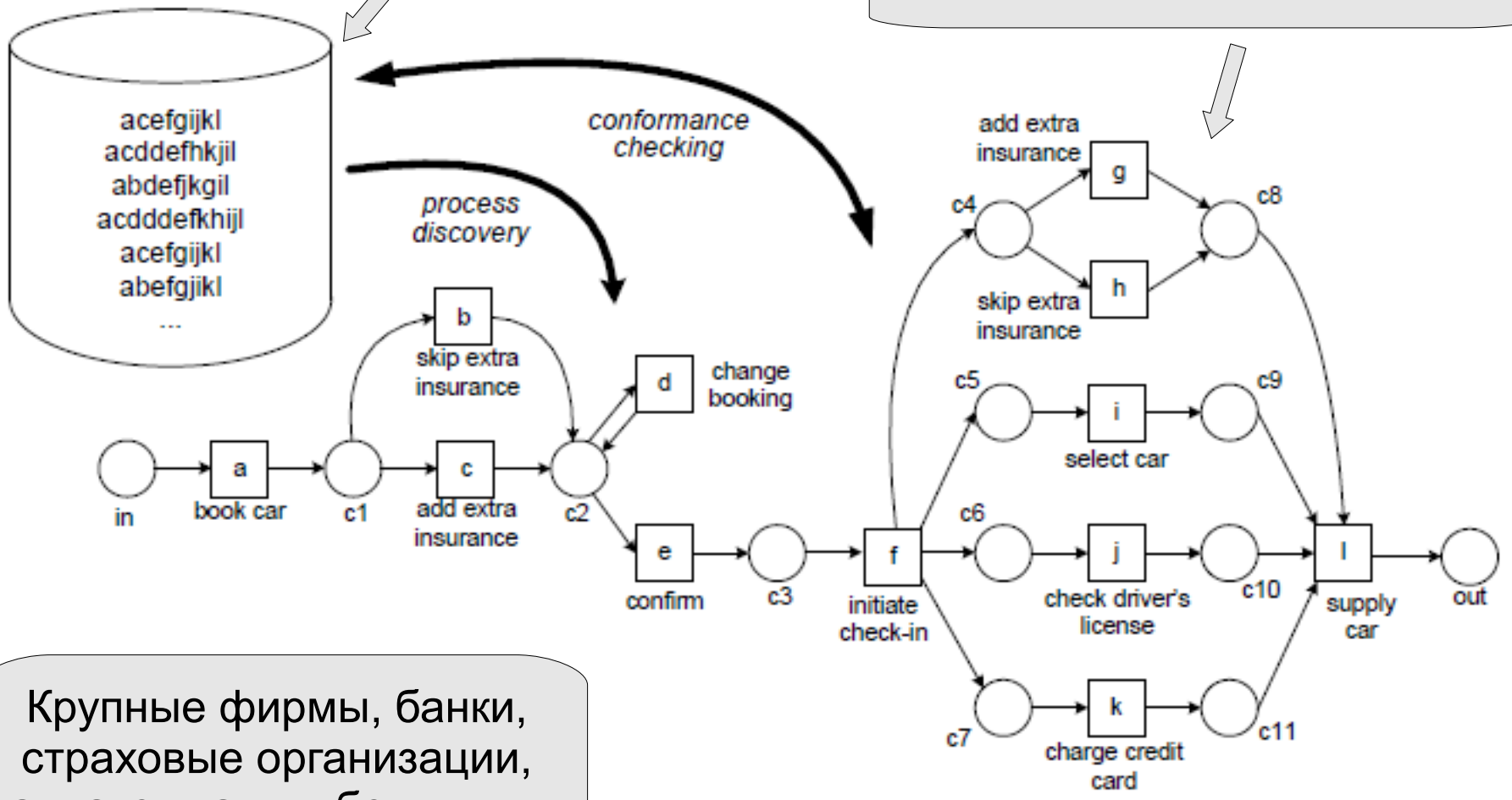
Процедуры извлечения и анализа процессов



Данные

миллионы случаев
десятки миллионов событий

тысячи активностей



Крупные фирмы, банки, страховые организации, авиакомпании, больницы, морские перевозчики

Распределенные вычисления

- Многоядерные системы (multicore computing)
- Многопроцессорные системы (manycore GPU)
- Кластерные системы (cluster computing)
- Грид-вычисления (grid computing)
- Облачные вычисления (cloud computing)
- ...

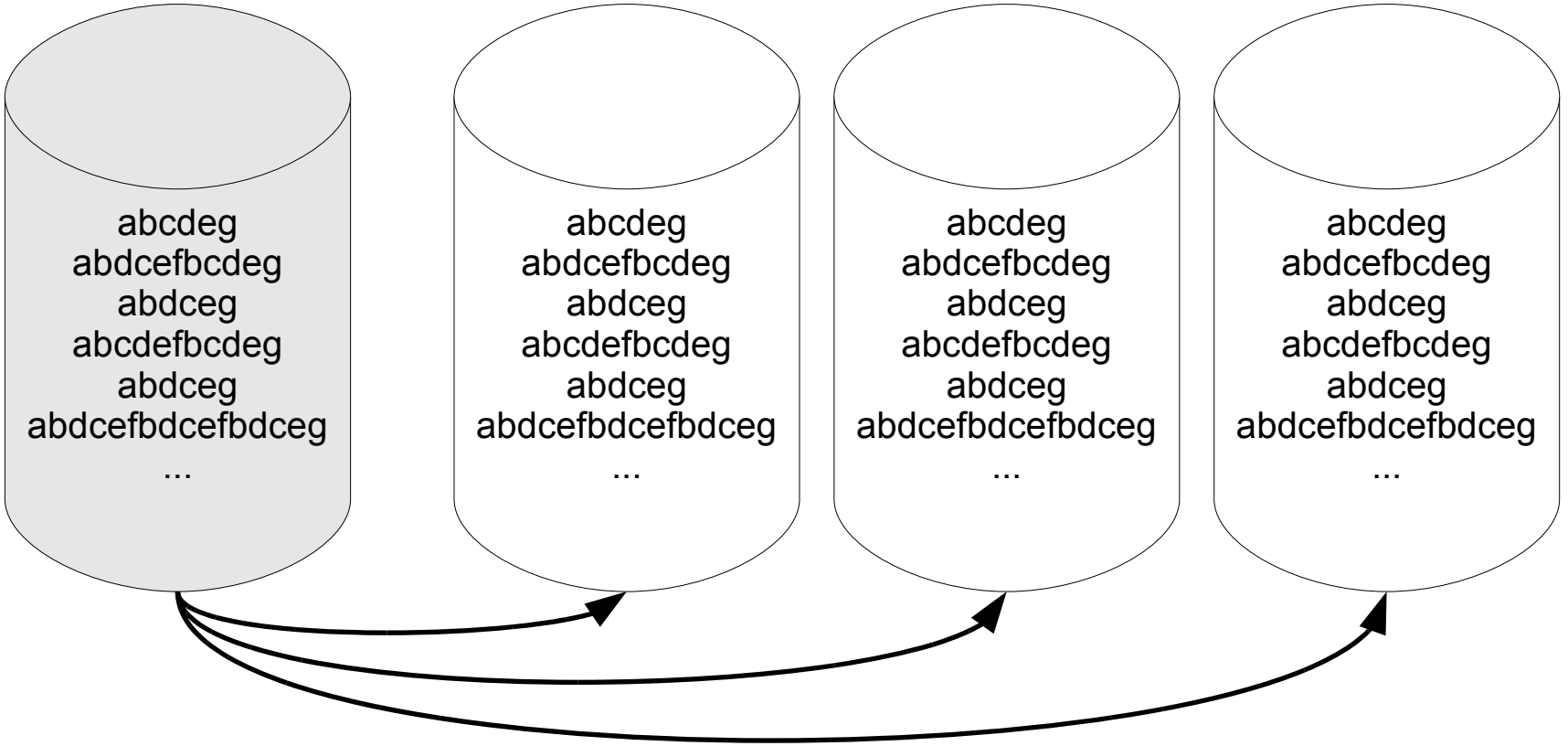
Количество транзисторов на одну схему постоянно растет, однако, для того, чтобы задействовать вычислительные мощности, предсказываемые законом Мура (экспоненциальный рост), необходимо использовать распределенные вычисления.

Общее количество данных в мире растет экспоненциально.

Hilbert, M., Lopez, P.: TheWorld's Technological Capacity to Store, Communicate, and Compute Information. Science 332(60) (2011)

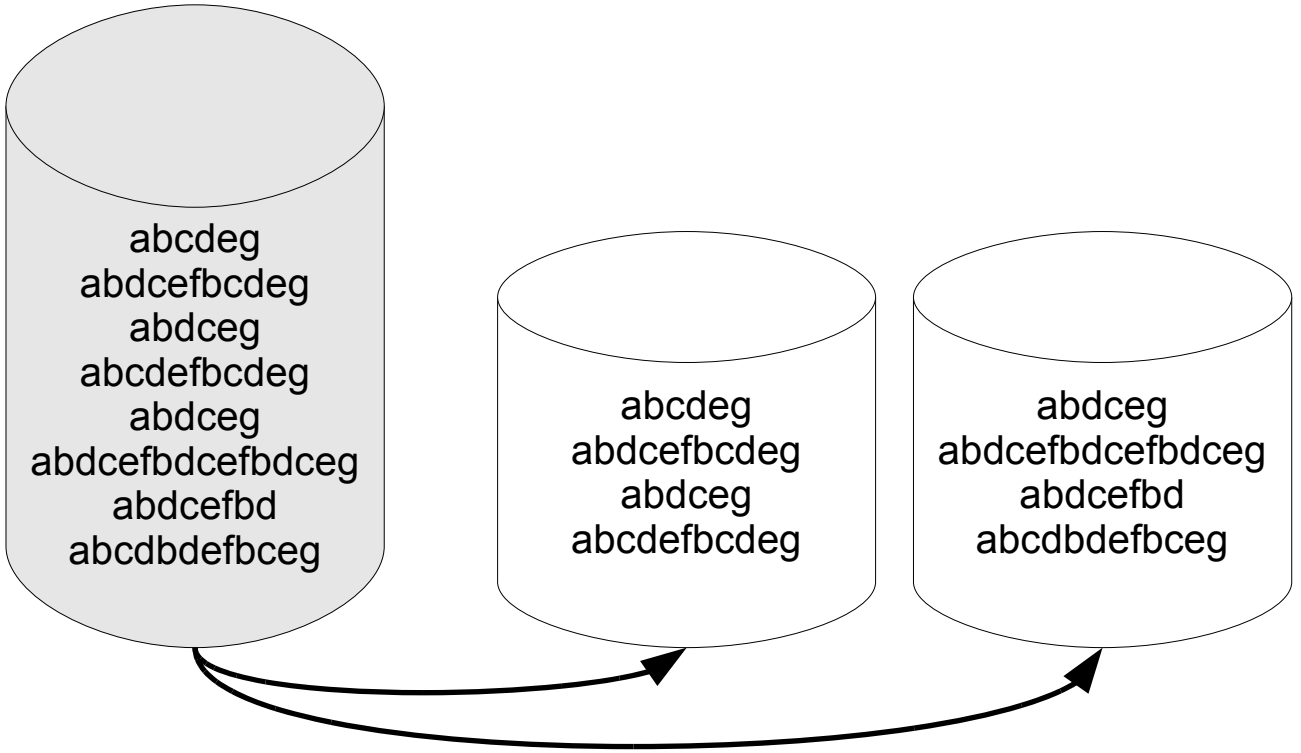
Три типа разбиения лога

Репликация | Вертикальное разбиение | Горизонтальное разбиение



Три типа разбиения лога

Репликация | **Вертикальное разбиение** | Горизонтальное разбиение



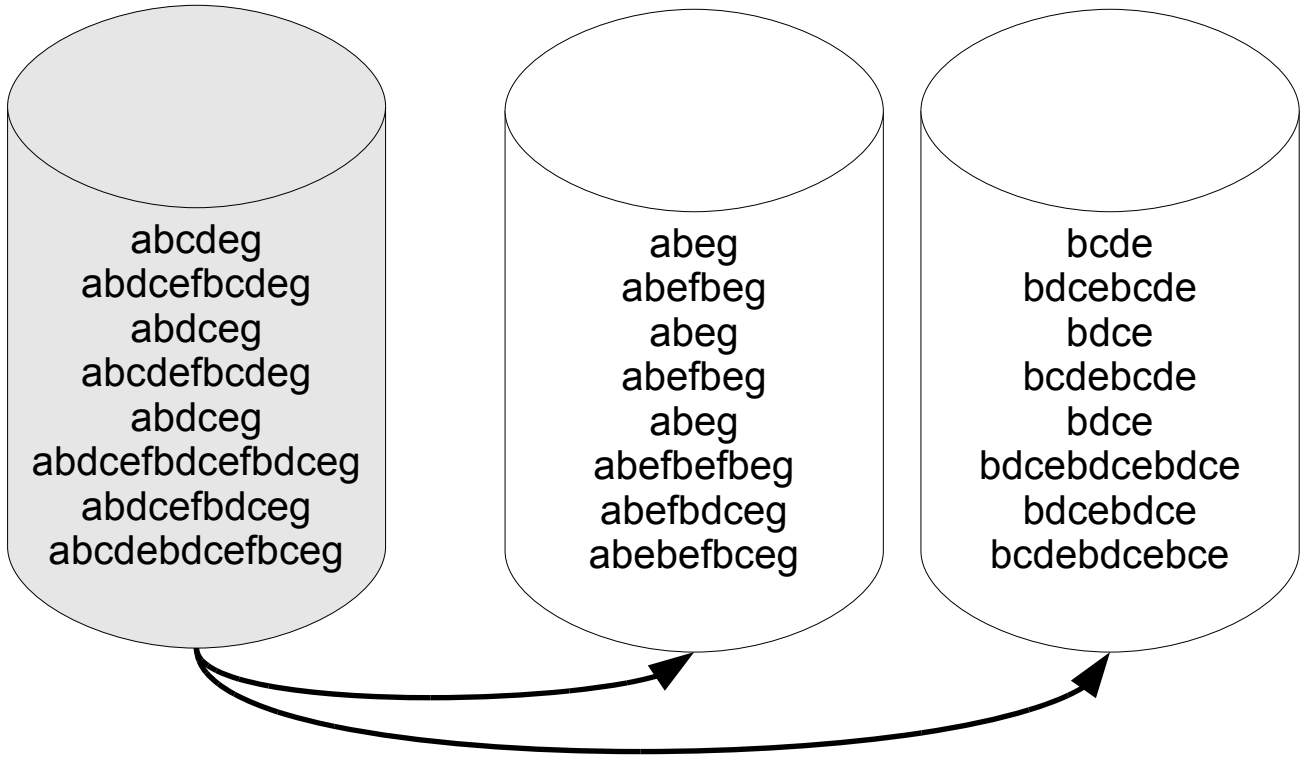
Возможны различные правила разбиения лога (concept drift).

Сложность процедуры объединения результатов вычислений на частях лога зависит от вида результата, который получается на выходе части (процент «хороших» случаев или полноценные модели процессов).

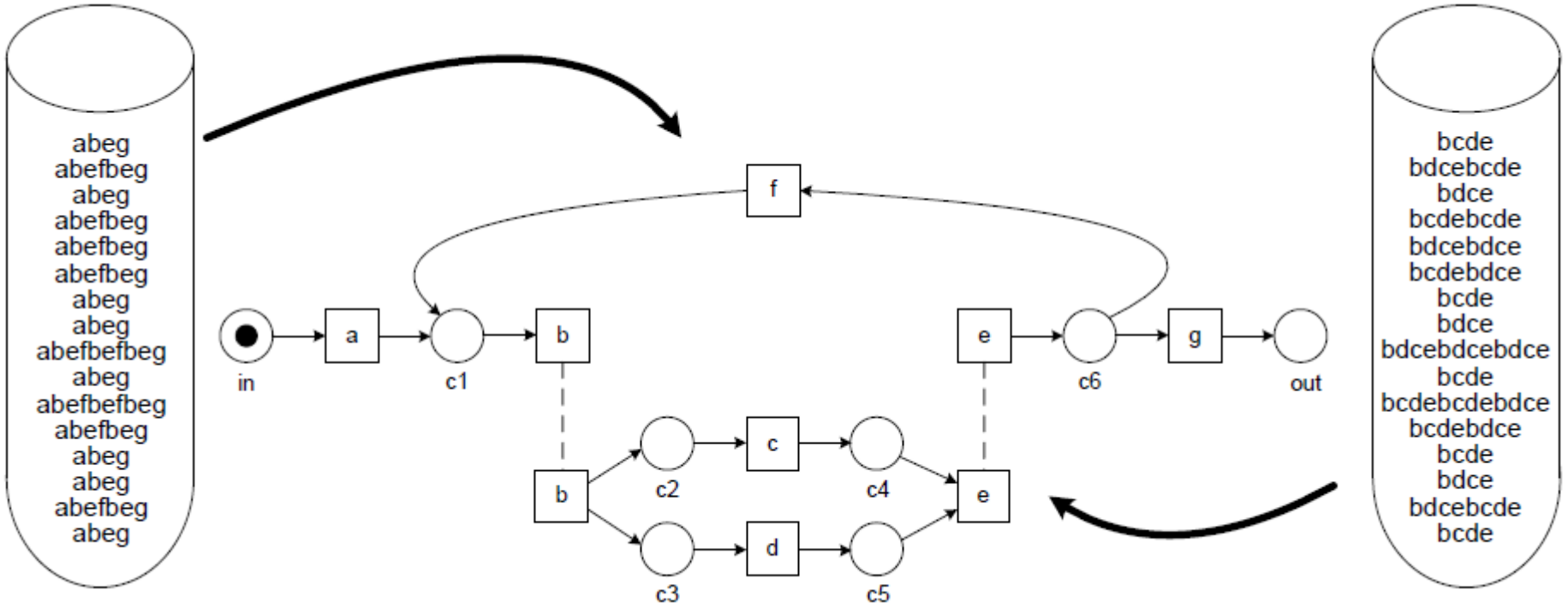
Чем более «низкоуровневый» выход, тем проще объединение.

Три типа разбиения лога

Репликация | Вертикальное разбиение | **Горизонтальное разбиение**

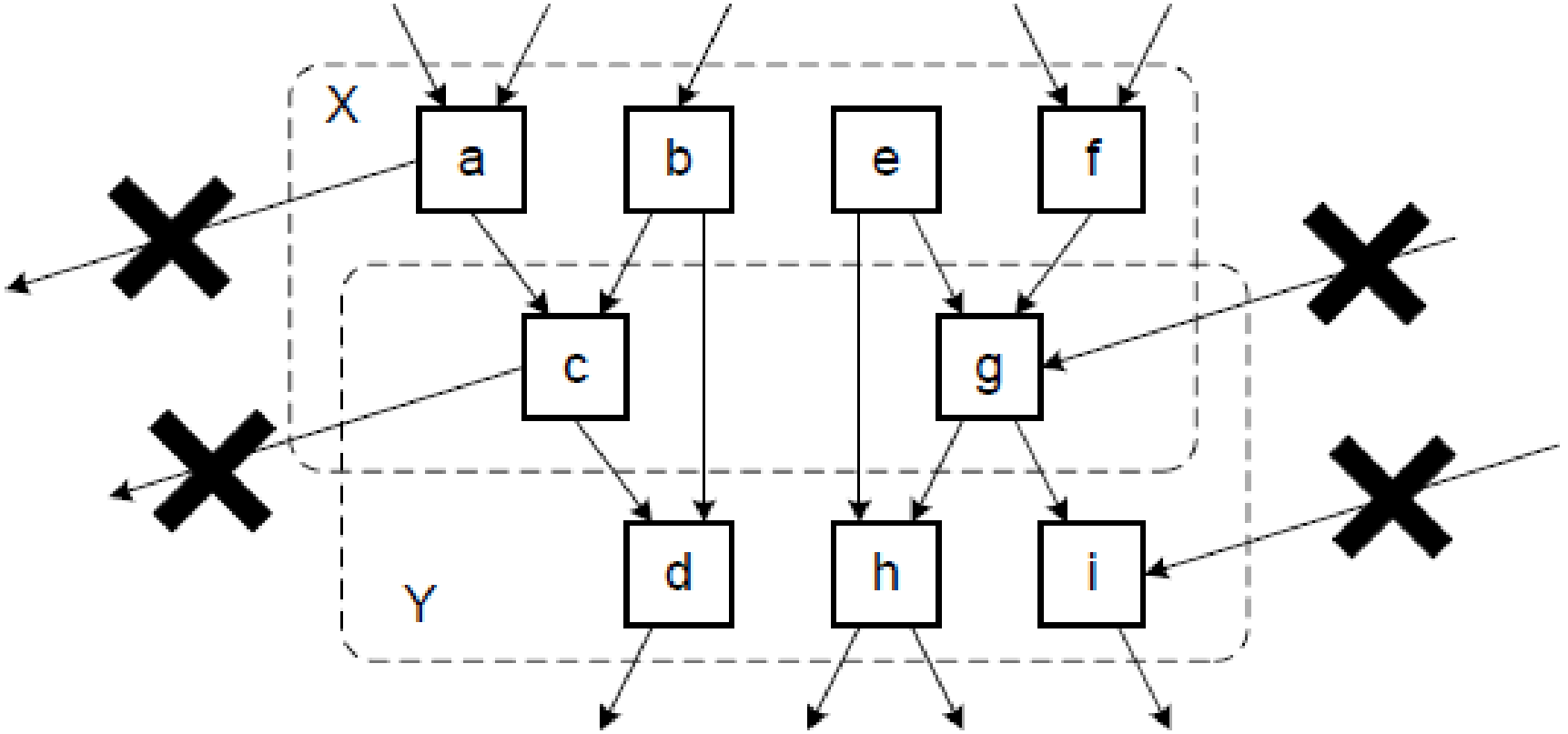


Горизонтальное разбиение: сборка результата

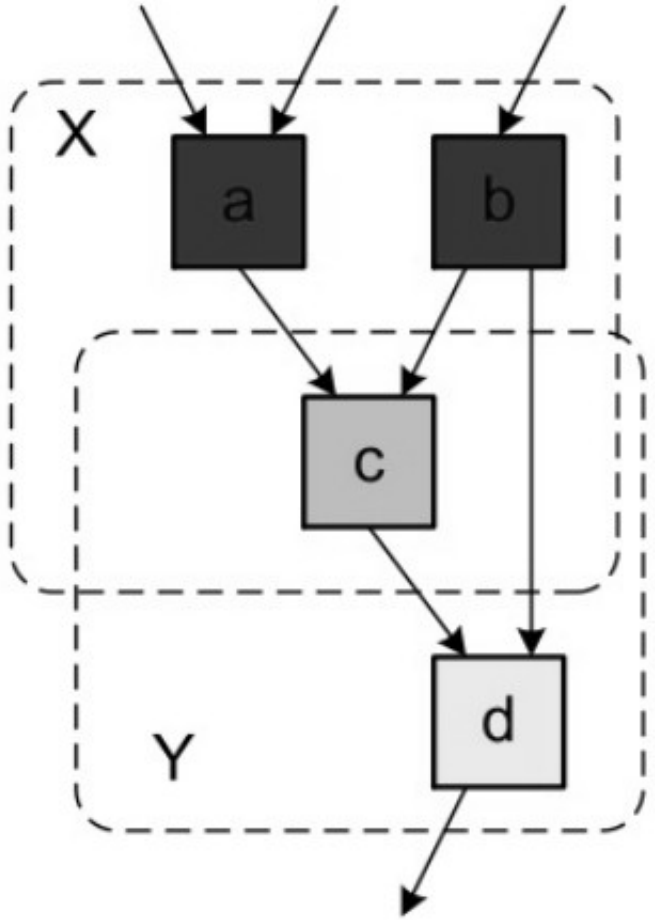


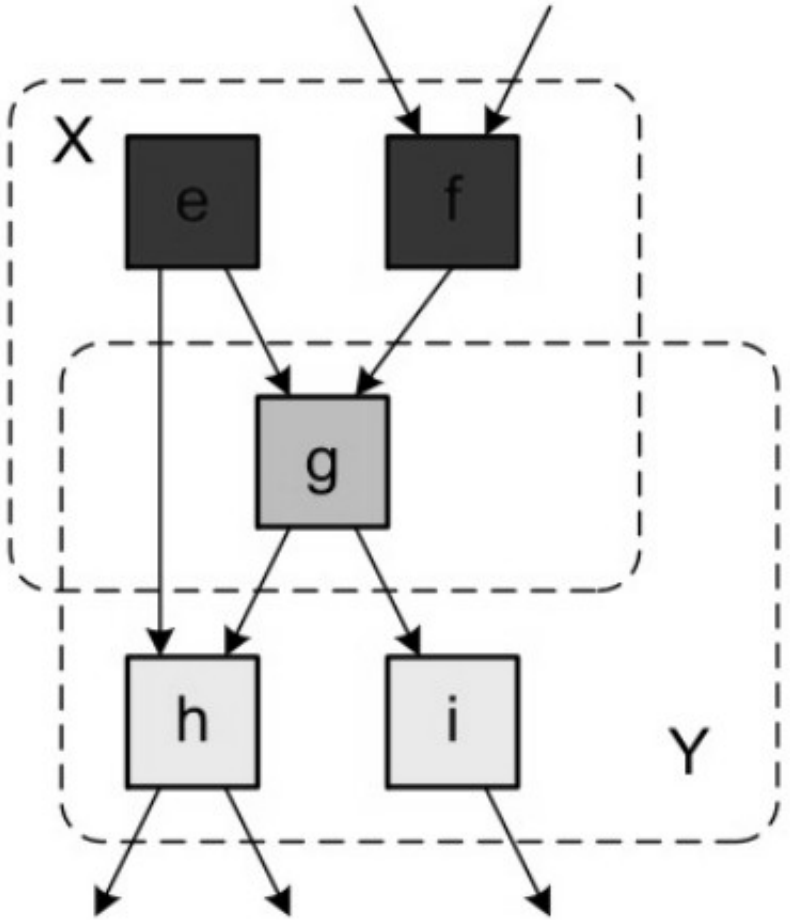
Пассажи (passages)

Определение (пассаж). G = (N,E) — граф.
P = (X, Y) — пассаж $\Leftrightarrow \varnothing \neq \mathbf{X} \subseteq \mathbf{N}$, $\varnothing \neq \mathbf{Y} \subseteq \mathbf{N}$, $\mathbf{X}\bullet = \mathbf{Y}$, $\mathbf{X} = \bullet\mathbf{Y}$.
pas(G) — множество пассажей на G.



Минимальные пассажи (minimal passages)

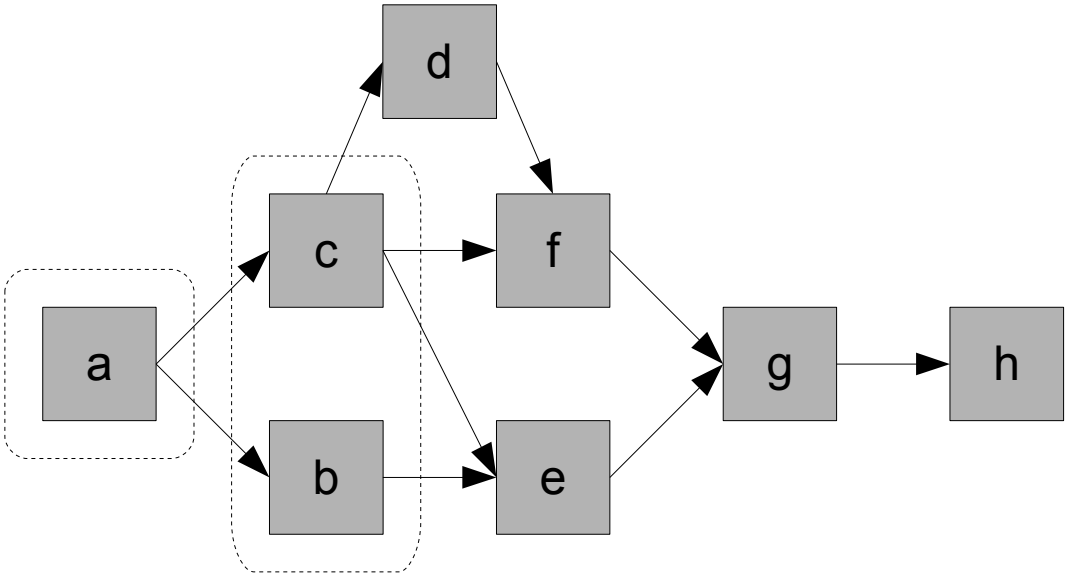




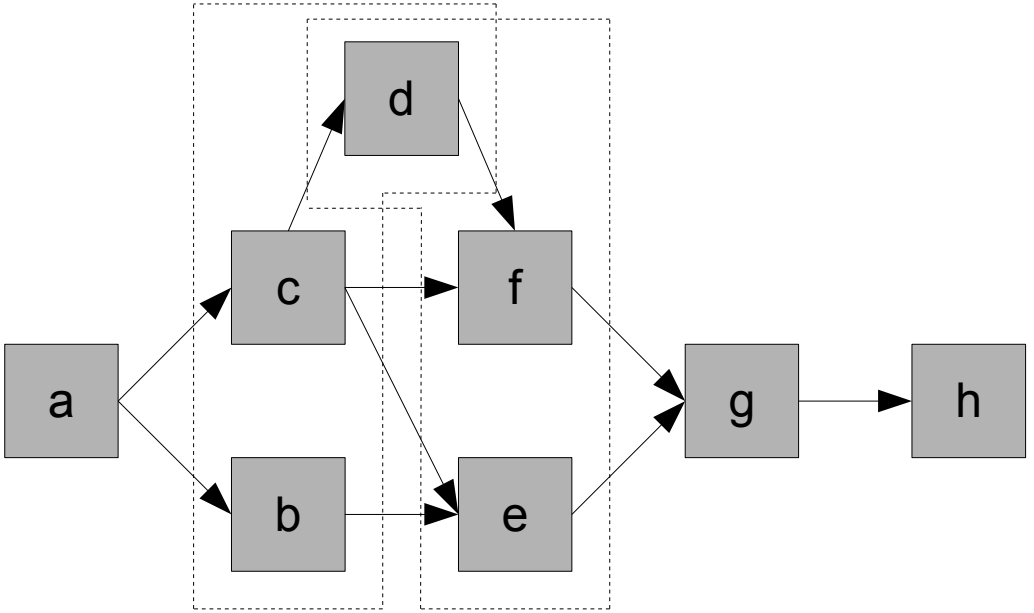
$P \in pas(G)$ — минимальный,
если нет такого $P' \in pas(G)$, что $P' < P$

Лемма. $G = (N,E)$ граф и $(x, y) \in E$.
Существует единственный минимальный пассаж $P(x,y) = (X,Y) \in pasmin(G)$ такой, что $x \in X$ и $y \in Y$.
Пассажи задают отношение эквивалентности на ребрах графа: $(x1,y1) \sim (x2,y2) \Leftrightarrow P(x1,y1) = P(x2,y2)$.

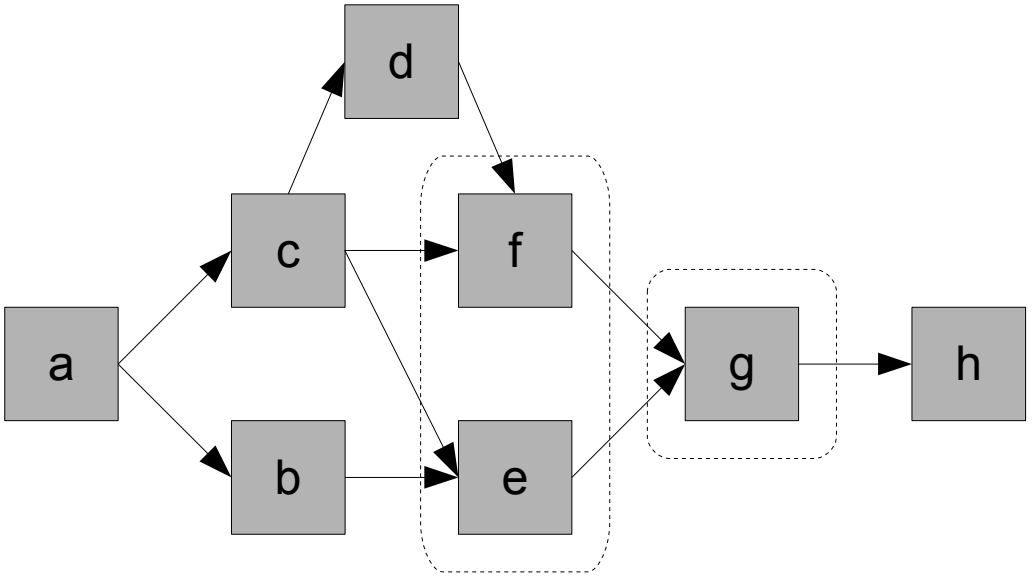
Passage ({a},{b,c})



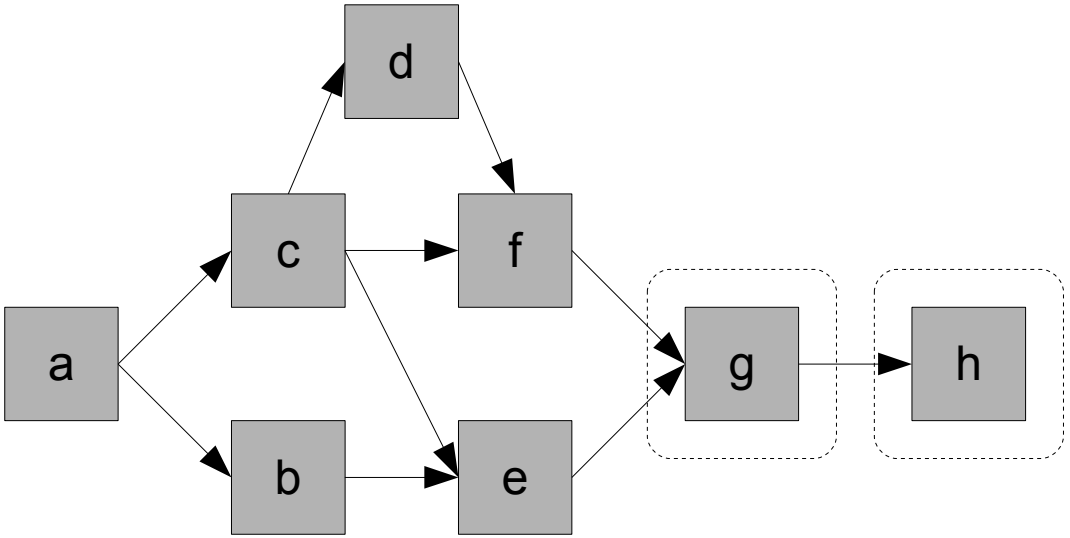
Passage ({b,c,d},{d,e,f})



Passage ({e,f},{g})



Passage ({g},{h})



1) Любой процесс можно разбить на подпроцессы, используя минимальные пассажи.

2) Процедуры process discovery и conformance checking можно производить для каждого подпроцесса отдельно.

Определения

Определение 1 (след, лог событий). A — множество. След $\sigma \in A^*$ — последовательность активностей.
Мультимножество следов $L \in B(A^*)$ — лог событий (event log).
Событие — активность + время события + ресурсы (дополнительные данные).

Определение 2 (сеть Петри). Тройка PN = (P, T, F), где P — множество позиций (place),
T — множество переходов (transition), $F \subseteq (P \times T) \cup (T \times P)$ — соединяющие их дуги (flow relation).

Определение 3 (разметка). PN = (P, T, F) — сеть Петри. Разметка M — мультимножество позиций: $M \in B(P)$.

Определение 4 (именованная сеть Петри). PN = (P, T, F, T_v), где (P, T, F) — сеть Петри,
$T_v \subseteq T$ — набор видимых меток.
$\sigma_v = <t1, t2, \ldots, tn> \in T_v^*$ - последовательность видимых переходов.
$M[\sigma_v \triangleright M'$ ⇔ существует последовательность $\sigma \in T^*$ такая, что $M[\sigma > M'$ и σ_v — проекция σ на T_v.

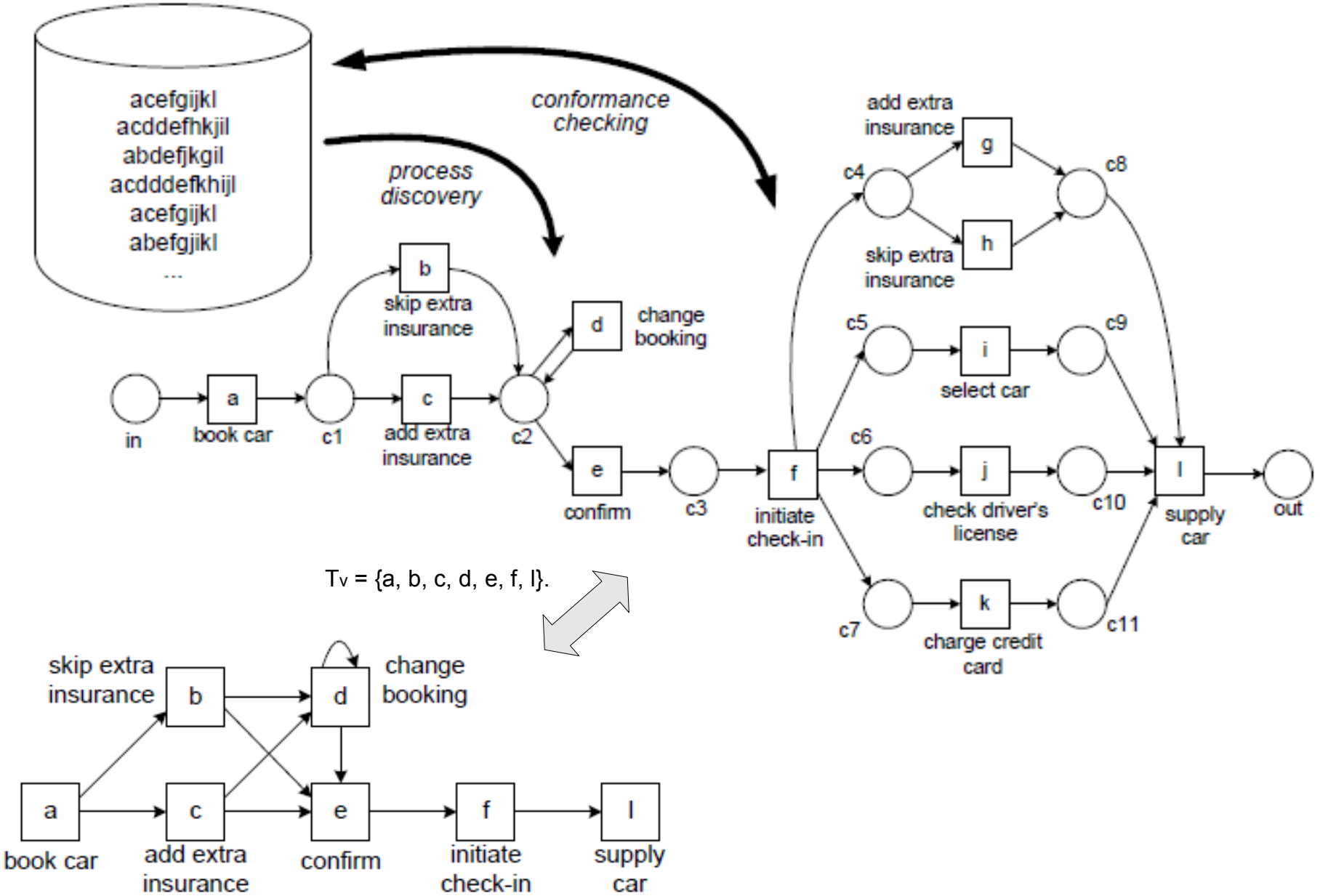
Определение 5 (WF-сеть). WF = (PN, in, Ti, out, To) — сеть потоков операций (workflow net), если
– PN = (P, T, F, T_v) — именованная сеть Петри,
– in ∈ P — исходная позиция такая, что ●in = ∅ и in● = T_i,
– out ∈ P — результирующая позиция такая, что out● = ∅ и ●out = T_o,
– $T_i \subseteq T_v$ — начальное множество переходов, ●T_i = {in},
– $T_o \subseteq T_v$ — конечное множество переходов, T_o● = {out},
– все узлы расположены на каком-либо пути из исходной в конечную позицию.

Определение 6 (путь). *G* = (*N,E*) — граф, *x, y* ∈ *N u Q* ⊆ *N.*

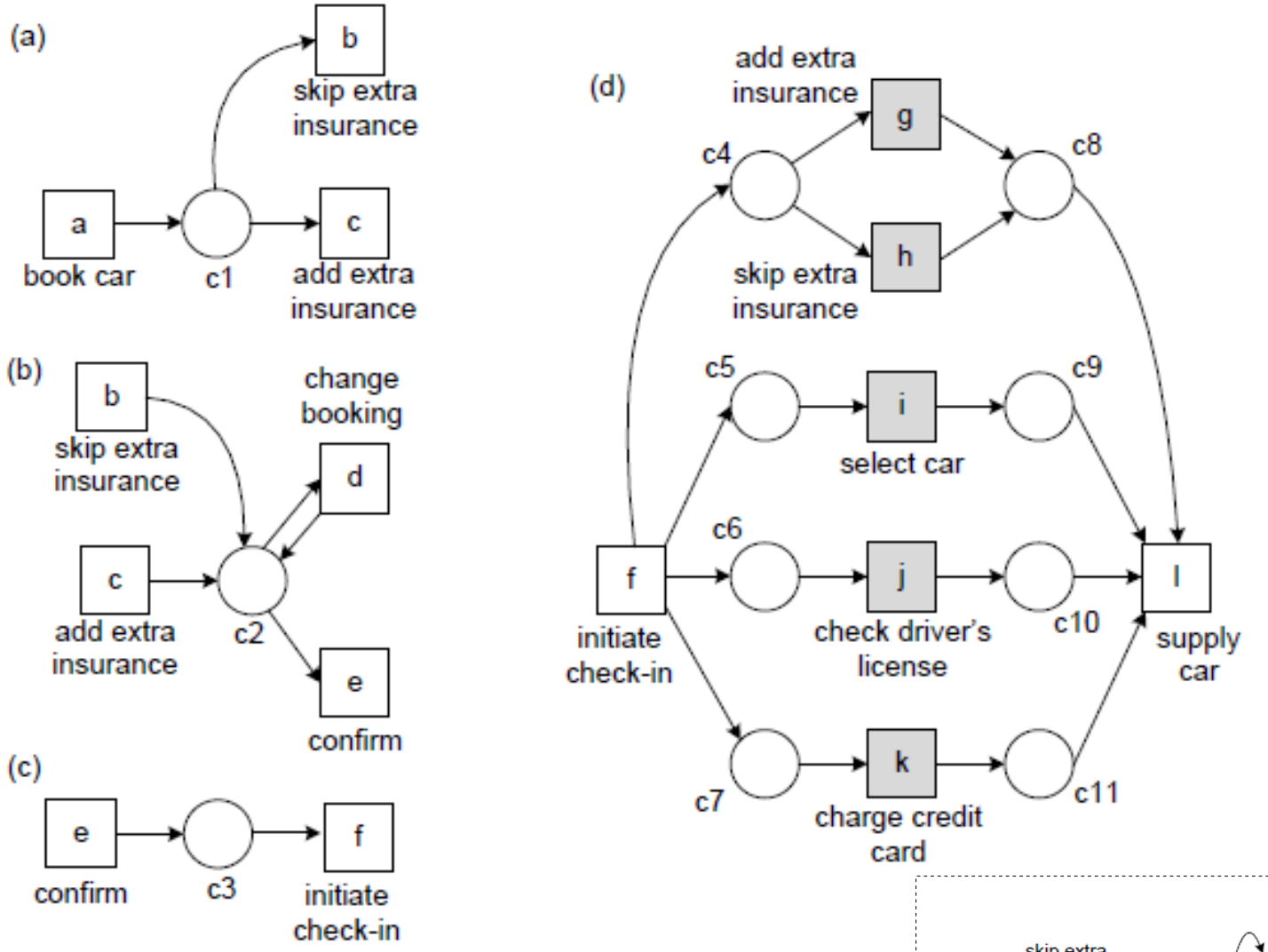
$x \overset{\sigma : E \# Q}{\rightsquigarrow} y$ ⇔ ∃ σ = <n_1, n_2, . . . n_k>, k > 1 такая, что x = n_1, y = n_k, ∀ 1 ≤ i < k: (n_i, n_{i+1}) ∈ E,
и ∀ 1 < i < k:$n_i \notin Q$.

Определение 7 (skeleton). PN = (P, T, F, T_v) — именованная сеть Петри.
Скелет сети PN — граф skel (PN) = (N,E), где N = T_v и E = {(x, y) ∈ $T_v \times T_v$ | $x \overset{F \# T_v}{\rightsquigarrow} y$ }.

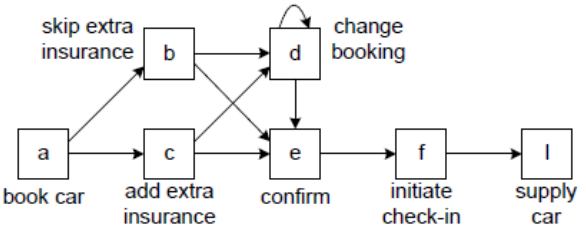
Пример: построение скелета



Пример: фрагменты сети Петри



Если в скелете есть несколько минимальных пассажей, то исходную сеть Петри можно разбить на фрагменты, соответствующие пассажам в скелете.



Пример: определения

Определение 8 (фрагмент сети). $PN = (P, T, F, T_v)$ — именованная сеть Петри.
Для любых двух наборов переходов $X, Y \subseteq T_v$ определим фрагмент сети $PN_{(X,Y)} = (P, T, F, T_v)$ так:

– $Z = \text{nodes}(X \overset{F\#T_v}{\rightsquigarrow} Y) \setminus (X \cup Y)$ — внутренние вершины фрагменты,
– $P = P \cap Z$,
– $T = (T \cap Z) \cup X \cup Y$,
– $F = F \cap ((P \times T) \cup (T \times P))$,
– $T_v = X \cup Y$.

Определение 9 (системная сеть). Тройка $SN = (PN, M_i, M_o)$, где
$PN = (P, T, F, T_v)$ — именованная сеть Петри с видимыми метками T_v,
$M_i \in B(P)$ — начальная разметка, $M_o \in B(P)$ — конечная разметка.

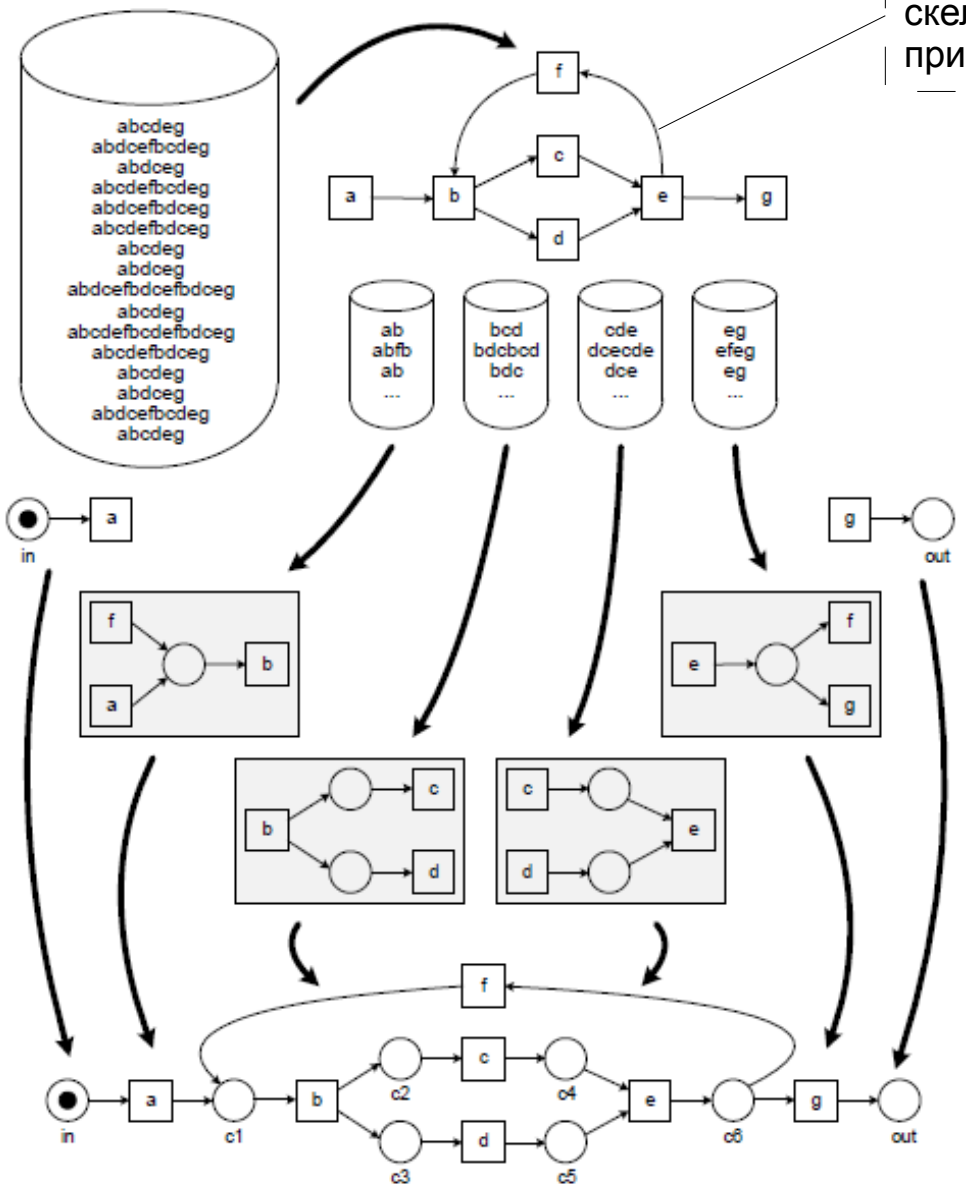
Определение 10 (идеальное соответствие). $L \in B(A^*)$ — лог событий, $SN = (PN, M_i, M_o)$ — системная сеть.
L идеально соответствует SN $\Leftrightarrow \{\sigma \in L\} \subseteq \tau(SN)$.

Теорема. $L \in B(A^*)$ — лог событий, $WF = (PN, in, T_i, out, T_o)$ — WF-сеть, где $PN = (P, T, F, T_v)$.
L идеально соответствует системной сети $SN = (PN, [in], [out])$ $\Leftrightarrow$
– $\forall\ a_1, a_2, \ldots a_k \in L: a_1 \in T_i$ и $a_k \in T_o$,
– $\forall\ (X, Y) \in pas_{min}(skel(PN))$: $L{\uparrow}_{X \cup Y}$ идеально соответствует $SN_{(X,Y)} = (PN_{(X,Y)}, [\,], [\,])$.

Если лог событий идеально соответствует всем пассажам,
то он соответствует и модели целиком. Обратное также верно.

Process discovery

Причинная структура (causal structure) — аналог скелета. Если удается ее получить, можно применить технику, рассмотренную ранее.



Разбивая одну большую проблему на много маленьких, получаем существенный прирост производительности.

Задачу можно решать в рамках широко распределенной вычислительной системы.

Большинство алгоритмов process mining имеют экспоненциально растущее с ростом количества активностей время работы. Использование распределенных алгоритмов оправдано даже на одном вычислителе (решать много «маленьких» задач дешевле, чем одну «большую»).

Эффективность распределенных алгоритмов сильно зависит от размера модели и лога событий. Чем больше активностей и случаев, тем больше прирост производительности.

Использование распределенных вычислений в process mining не привязано к конкретным алгоритмам и нотациям.