Такива символи логиците наричат предикатни символи.

Да дадем и точна дефиниция за видовете символи, които разгледахме.

✓ **2.4.1. Дефиниция.** а) Да фиксираме от тук до края на този курс два сорта **индив** и **съжд**.

б) Променливите от сорт **индив** се наричат *индивидни променливи*.

в) Символите, чийто тип е

$$\langle\rangle \mapsto \textbf{индив}$$

се наричат *символи за индивидни константи*.

г) Символите, чийто тип е

$$\langle \underbrace{\textbf{индив}, \textbf{индив}, \ldots, \textbf{индив}}_{n \text{ броя } \textbf{индив}} \rangle \mapsto \textbf{индив}$$

се наричат *n-местни функционални символи*.

д) Символите, чийто тип е

$$\langle \underbrace{\textbf{индив}, \textbf{индив}, \ldots, \textbf{индив}}_{n \text{ броя } \textbf{индив}} \rangle \mapsto \textbf{съжд}$$

се наричат *n-местни предикатни символи*.

Забележка: Важно е да не забравяме, че символите за индивидни константи, функционалните символи и предикатните символи са символи, които сами по себе си нямат фиксиран смисъл. Също както π не е число, а буква от гръцката азбука, с която означаваме определено число, така и символът „Солон“ не е цар на Атѝна, а означение, което можем да използваме, за да назоваваме този цар. Обаче нищо не пречи да използваме означението „Солон“ и за напълно различен индивид, например за числото 732691.

Тъй като в предикатната логика няма съждителни променливи, то за краткост, вместо „индивидна променлива“, може да казваме просто *променлива*.

Също така, вместо термина „символ за индивидна константа“ може да използваме и по-кратките изрази *символ за константа*, *индивидна константа* и дори само *константа*. Това е възможно, тъй като в контекста на логическото програмиране и математическата логика други константи почти няма и значи винаги когато кажем, че нещо е „константа“, е ясно, че всъщност искаме да кажем, че то е „символ за индивидна

константа“. В това отношение функционалните и предикатните символи са „онеправдани“, защото в логическото програмиране се използват функции и предикати и затова не е правилно вместо „функционален символ“ да казваме накратко „функция“, нито пък вместо „предикатен символ“ може да казваме „предикат“.*

Някои математици считат, че символите за индивидни константи всъщност са специален вид функционални символи, а именно — нулместни функционални символи (т. е. с нула аргументи). При други математици пък функционалните символи задължително имат поне един аргумент и значи символите за индивидни константи не са функционални символи.

За константите (или нулместните функционални символи), както и за нулместните предикатни символи казваме, че са *нуларни*. За едноместните функционални и предикатни символи казваме, че са *унарни*. За двуместните функционални и предикатни символи казваме, че са *бинарни*. За триместните функционални и предикатни символи казваме, че са *тернарни*. Изобщо, за n-местните функционални и предикатни символи казваме, че са n-арни, а n е тяхната *арност*** или *местност*.

✓ **2.4.2. Уговорка.** За да не се налага винаги изрично да се уточнява кой символ е константа, кой функционален, кой предикатен и кой променлива, в математиката се използват следните уговорки:

- за променливи се използват буквите $\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{t}, \mathtt{u}, \mathtt{v}, \mathtt{w}$;
- буквите $\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}$ са символи за индивидни константи;
- за функционални (не нулместни) символи се използват буквите $\mathtt{f}, \mathtt{g}, \mathtt{h}$, а при нужда още и $\mathtt{j}, \mathtt{k}, \mathtt{l}$.
- за предикатни символи се използват буквите $\mathtt{p}$, $\mathtt{q}$, $\mathtt{r}$.

Ако трябват повече символи, се използват индекси, примове и т. н., напр. $\mathtt{x}$, $\mathtt{x}'$, $\mathtt{x}_3$ и $\mathtt{x}''_2$ са променливи, а $\mathtt{p}$, $\mathtt{p}_1$ и $\mathtt{p}''_2$ са предикатни символи.

Благодарение на тази уговорка, когато видим израз като $\mathtt{f}(\mathtt{g}(\mathtt{c}), \mathtt{x}, \mathtt{b})$, е ясно, че това е терм, а не атомарна формула (защо?). В този терм $\mathtt{f}$ е

*На изпита по логическо програмиране се счита за груба грешка, ако за нещо, което е функционален символ, се каже че е функция.

Този смисъл на думата „арност“ се различава от дадения в дефиниция 2.1.2 г), според който арността на например един двуместен функционален символ би била не числото 2, а двойката $\langle$индив**, **индив**$\rangle$. Тази двойна употреба не води до проблеми, защото в едносортната предикатна логика сортовете на аргументите винаги са **индив**. Затова ако например кажем, че арността е 2, от тук е ясно, че другият вид арност е $\langle$**индив**, **индив**$\rangle$. И обратно, ако кажем, че арността е $\langle$**индив**, **индив**$\rangle$, то от това става ясно, че числовата арност е 2.

триместен функционален символ, g е едноместен функционален символ, c и b са символи за константи и x е променлива. От друга страна изразът p(f(g(x), c)) е атомарна формула, а не терм. В нея p е едноместен предикатен символ, c е символ за константа, f е двуместен функционален символ, а g е едноместен функционален символ.

✓ ***Задача* 7:** Открийте функционалните символи, константите и променливите в терма f(f(c, a), f(g(g(x)), h(y))). За всеки от функционалните символи определете неговата арност.

✓ ***Задача* 8:** Кои от следните изрази не са термове и защо: f(c), c(f), c, f, x(c), f(f(x)), f(f(f(c))), f(f(f(c, c))), g(g(x, x), g(x, y)).

✓ ***Задача* 9:** Кои от следните изрази са атомарни формули и кои от тях не съдържат променливи: c, f(c), p(c), p(f(f(f(c)))), p(f(x)), f(p(x)), p(p(x)), f(f(x))?

Сигнатури

Използвайки символите, които дефинирахме в предния подраздел, можем да дадем точна дефиниция и на понятията терм и предикатна формула. За много приложения обаче е удобно да ограничим термовете и формулите да не използват произволни символи, а само символи, включени в определен от нас списък. Ще наречем този списък сигнатура.

2.4.3. Дефиниция. Крайна или безкрайна редица от символи за индивидни константи, функционални символи и предикатни символи се нарича *сигнатура*.*

В зависимост от това какви символи включим в сигнатурата, ще получим най-различни, нееквивалентни по между си варианти на предикатната логика. При някои сигнатури например може да няма функционални символи, а при други да има. При някои има само един предикатен символ, който е едноместен, при други само един, който е двуместен, при трети имаме безброй много триместни предикатни символи и т. н. Изразителната сила на предикатната логика е различна при всяка една от тези сигнатури. Ще използваме фрази като „терм τ при

*Терминът сигнатура (на англ. signature) се използва в математическата логика, алгебрата, теория на езиците за програмиране и информатиката. Понякога в математическата логика и алгебрата вместо за сигнатура се говори за *език* (language). В универсалната алгебра сигнатурите понякога се наричат *типове* (types). В теория на моделите пък се използва терминът *речник* (vocabulary).

сигнатура $\mathfrak{sig}$“ или „$\mathfrak{sig}$-терм τ“ и „атомарна формула φ при сигнатура $\mathfrak{sig}$“ или „$\mathfrak{sig}$-формула φ“, за да кажем, че термът τ и атомарната формула φ са построени, използвайки символите, предоставени от сигнатурата $\mathfrak{sig}$.

За да не усложняваме излишно формулировките на дефинициите и твърденията, нека фиксираме една конкретна сигнатура $\mathfrak{sig}$. Ако в някоя дефиниция или твърдение говорим просто за „функционални символи“, „променливи“, „термове“ и т. н., без да е споменато изобщо за сигнатура, то ще имаме предвид функционални символи, променливи и термове при така фиксираната сигнатура $\mathfrak{sig}$.

Термове

Вече сме готови да дадем и дефиницията на терм. Дефиницията ще бъде индуктивна. Това означава, че ако за даден израз успеем да докажем, че е терм, използвайки в доказателството единствено трите правила от дефиниция 2.4.4, то тогава този израз е терм. Ако пък за даден израз е невъзможно да се докаже, че е терм, използвайки единствено тези три правила, то тогава той не е терм.

✓ **2.4.4. Дефиниция.** Нека $\mathfrak{sig}$ е сигнатура. Понятието *терм* при сигнатура $\mathfrak{sig}$ или $\mathfrak{sig}$-терм се дефинира индуктивно посредством следните правила:

а) Ако $\mathtt{x}$ е индивидна променлива, то $\mathtt{x}$ е $\mathfrak{sig}$-терм.

б) Ако $\mathtt{c}$ е символ за константа от $\mathfrak{sig}$, то $\mathtt{c}$ е $\mathfrak{sig}$-терм.

в) Ако $\mathtt{f}$ е n-местен функционален символ от $\mathfrak{sig}$ и $\tau_1, \tau_2, \ldots, \tau_n$ са $\mathfrak{sig}$-термове, то низът $\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$ е $\mathfrak{sig}$-терм.

2.4.5. ***Забележка:*** Вижда се, че в така дадената дефиниция използваме традиционен функционален запис за термовете (вж. пример 2.1.10). Нямаше да има никакъв проблем в нея да използваме кой да е друг запис, който гарантира еднозначност на синтактичния разбор. Например ако искахме да ползваме термове в полски запис, в 2.4.4 в) вместо за $\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$ можеше да кажем за низа $\mathtt{f}\tau_1\tau_2\ldots\tau_n$, че е $\mathfrak{sig}$-терм.

✓ ***Задача* 10:** Нека $\mathtt{x}$ е променлива, а сигнатурата $\mathfrak{sig}$ включва символ за константа $\mathtt{c}$ и двуместен функционален символ $\mathtt{f}$. Използвайки правилата от дефиниция 2.4.4, докажете, че $\mathtt{f}(\mathtt{c}, \mathtt{f}(\mathtt{c}, \mathtt{x}))$ е $\mathfrak{sig}$-терм.

***Задача* 11:** Съществуват ли термове:

– които съдържат скоби, но не съдържат запетаи?

– за по-добра четливост бихме позволявали използването на интервали в термове от вида $\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$;

– в някои случаи бихме допускали инфиксен запис, напр. $\mathtt{a}+\mathtt{b}$ вместо $+(\mathtt{a}, \mathtt{b})$.

Но въпреки че нашата дефиниция не споменава изрично всички тези „удобства“, няма никакви причини да не се възползваме от тях. Например, спокойно може да пишем термове като $\mathtt{a} + \mathtt{b}$, стига да се сещаме, че това всъщност е термът $+(\mathtt{a}, \mathtt{b})$.

✓ ***Задача* 14:** Колко скоби се съдържат в терма $\mathtt{b} + \mathtt{x} * \mathtt{a}$?

***Задача* 15:** Нека сигнатурата $\mathfrak{sig}$ е такава, че в нея няма нито един символ за константа. Използвайки индуктивния принцип за термове, докажете, че всеки $\mathfrak{sig}$-терм съдържа поне една променлива.

***Задача* 16:** Нека сигнатурата $\mathfrak{sig}$ е такава, че всички функционални символи от $\mathfrak{sig}$ имат арност 2 (или 0). Използвайки индуктивния принцип за термове, докажете, че във всички термове при сигнатура $\mathfrak{sig}$ броят на запетаите е равен на броя на десните скоби.

Атомарни формули

Атомарните формули при сигнатура $\mathfrak{sig}$ може да дефинираме така:

✓ **2.4.6. Дефиниция.** Ако $\mathtt{p}$ е n-местен предикатен символ от $\mathfrak{sig}$ и $\tau_1, \tau_2, \ldots, \tau_n$ са $\mathfrak{sig}$-термове, то низът $\mathtt{p}(\tau_1, \tau_2, \ldots, \tau_n)$ е *атомарна формула* при сигнатура $\mathfrak{sig}$ (или атомарна $\mathfrak{sig}$-формула).

2.4.7. Означение. За удобство атомарните формули, образувани от нулместен предикатен символ, обикновено се записват без скоби, т.е. $\mathtt{p}$, а не $\mathtt{p}()$. Това е аналогично на термовете, образувани от нулместни функционални символи, т.е. символи за константи. Например ако $\mathtt{c}$ е символ за константа, то термът, образуван от $\mathtt{c}$ е просто $\mathtt{c}$, а не $\mathtt{c}()$.*

Просто сравнение на тази дефиниция с дефиницията на терм (вж. точка 2.4.4 в) от дефиницията) показва, че една атомарна формула $\mathtt{p}(\tau_1, \tau_2, \ldots, \tau_n)$ прилича по всичко на терм с единствената разлика, че

*Причината, поради която дефиниция 2.4.6 не е съобразена с този начин за запис, е това, че така си спестяваме необходимостта в различните доказателства да разглеждаме случаи според арността на предикатните символи. Впрочем тук не сме напълно последователни, защото ако в дефиницията за терм не бяхме отделили символите за константи като отделен случай, а считахме, че те са нулместни функционални символи, то бихме си опростили някои от доказателствата още повече.

$\mathtt{p}$ не е функционален, а предикатен символ. И също както при термовете, когато например + е двуместен функционален символ, се уговорихме за удобство да използваме инфиксен запис като пишем например $\mathtt{a}+\mathtt{b}$ вместо $+(\mathtt{a},\mathtt{b})$, така и при атомарните формули при някои предикатни символи за удобство ще използваме инфиксен запис. Например ако „$<$“ е двуместен предикатен символ, ще пишем $\mathtt{a}<\mathtt{b}$ вместо $<(\mathtt{a},\mathtt{b})$ и също ако „$=$“ е двуместен предикатен символ, ще пишем $\mathtt{a}=\mathtt{b}$ вместо $=(\mathtt{a},\mathtt{b})$. Разбира се, това правим само за наше удобство. Правилните атомарни формули, отговарящи на дефиниция 2.4.6, са $<(\mathtt{a},\mathtt{b})$ и $=(\mathtt{a},\mathtt{b})$.

Тъй като атомарните формули приличат на термове, може да възникне въпросът защо изобщо имаме нужда от това понятие. Не може ли да считаме символите $=$ и $<$ за функционални, при което $\mathtt{a}=\mathtt{b}$ и $\mathtt{a}<\mathtt{b}$ ще станат термове и няма да имаме нужда от атомарни формули?

Отговорът на този въпрос е следният: въпреки че като запис термовете и атомарните формули си приличат, сортовете ги различават. Термовете са синтактични обекти, чиято стойност е от сорт **индив**, а атомарните формули — синтактични обекти, чиято стойност е от сорт **съжд**. Например може да сложим терм като аргумент на функционален символ и тогава получаваме по-голям терм. Също може да сложим термове като аргумент на предикатен символ и в резултат получаваме атомарна формула. Обаче ако сложим атомарна формули като аргумент на функционален или предикатен символ, резултатът е безсмислица.

✓ **2.4.8. Пример.** Ето примерен терм, който се получава като дадем по-малкия терм „Зевс“ като аргумент на функционалния символ „баща_на“:

баща_на(Зевс)

Ето примерна атомарна формула, която се получава като дадем термовете „баща_на(Зевс)“ и „Прометей“ като аргументи на двуместния предикатен символ „чичо“:

чичо(баща_на(Зевс), Прометей)

Ето примерна безсмислица, която се получава като дадем атомарна формула като аргумент на функционален символ:

баща_на(чичо(Кронос, Прометей))

Интуитивно атомарната формула „чичо(Кронос, Прометей)“ казва, че Кронос е чичо на Прометей. Т.е. това е някакво съждение. Кой е бащата на това съждение? Разбира се, това е безсмислен въпрос. Съжденията могат да бъдат например верни и могат също да бъдат неверни, но

нямат бащи. Ето още една безсмислица, която се получава като дадем атомарна формула като аргумент на предикатен символ:

чичо(Хиперион, чичо(Кронос, Прометей))

Чий чичо е Хиперион? На съждението „Кронос е чичо на Прометей“.

✓ **2.4.9. Пример.** Да разгледаме следните три израза, в които 0 е символ за константа, „+“ е двуместен функционален символ и = е двуместен предикатен символ:

$$0 + (0 + 0)$$
$$0 = (0 + 0)$$
$$0 + (0 = 0)$$

Първият от тези изрази е терм. Например ако интерпретираме символа 0 като числото нула, а „+“ като събиране на числа, тогава стойността на този терм е нула. Вторият от тези изрази е атомарна формула. Ако отново интерпретираме символа 0 като числото нула, „+“ като събиране на числа и „=“ като равенство, тогава тази атомарна формула е вярна. Третият от горните изрази обаче е безсмислица. Не може да съберем $0 = 0$ с числото нула.*

Предикатни формули

В раздел 2.3 вече видяхме, че много съждения не могат да се изразят, използвайки само средствата, с които разполага съждителната логика. Например съждения като „Сократ е смъртен“ и „Орфей обича Евридика“ е най-естествено да се запишат като атомарни формули. Обаче не може да се ограничим само с атомарните формули, защото много съждения се формулират, използвайки различни логически операции, а в атомарните формули няма логически операции. Да разгледаме например следното съждение:

Ако Афродита е най-красивата, то Елена ще обича Парис.

Използвайки средствата на съждителната логика, можем да представим това съждение посредством формулата $\mathtt{x} \Rightarrow \mathtt{y}$, където $\mathtt{x}$ е съждителна променлива, с която означаваме съждението „Афродита е най-красивата“, а $\mathtt{y}$ съждителна променлива, с която означаваме съждението „Елена обича Парис“. Съждителните променливи $\mathtt{x}$ и $\mathtt{y}$ обаче по

*В някои езици за програмиране, напр. си, този израз не е безсмислен (по-точно изразът `0+(0==0)`). Това е така само защото в тези езици равенството не е логическа, а аритметична операция, чиято стойност е числото едно или нула, а не истина или лъжа.

✓ **2.4.10. Дефиниция.** *Предикатните формулите* при сигнатура $\mathfrak{sig}$ или просто $\mathfrak{sig}$-*формулите* се дефинират индуктивно:

а) ако φ е атомарна $\mathfrak{sig}$-формула, то φ е $\mathfrak{sig}$-формула;

б) $\bot$ е $\mathfrak{sig}$-формула;

в) ако φ и ψ са $\mathfrak{sig}$-формули, то низовете $(\varphi \,\&\, \psi)$, $(\varphi \vee \psi)$ и $(\varphi \Rightarrow \psi)$ са $\mathfrak{sig}$-формули;

г) ако $\mathtt{x}$ е променлива и φ е $\mathfrak{sig}$-формула, то низовете $\forall \mathtt{x}\, \varphi$ и $\exists \mathtt{x}\, \varphi$ са $\mathfrak{sig}$-формули.

Разбира се, при различните сигнатури имаме различни атомарни формули, а значи при различните сигнатури ще имаме и различни формули. Множеството от всички формули при някоя конкретна сигнатура наричаме *език на предикатната логика* или просто *език*.

2.4.11. Когато се дава точната дефиниция на понятието „формула“, не е нужно в нея да се споменават изрично всички логически операции. Например в дефиниция 2.4.10 не сме казали нищо за операцията еквиваленция. Това не създава проблеми, защото можем да считаме, че всяка формула от вида $\varphi \Leftrightarrow \psi$ представлява съкратен запис на формулата $((\varphi \Rightarrow \psi) \,\&\, (\psi \Rightarrow \varphi))$. Също така не сме споменали и отрицанието. Това също не създава проблеми, защото може да считаме, че всяка формула от вида $\neg\varphi$ представлява съкратен запис на формулата $(\varphi \Rightarrow \bot)$. Удобно е също така да считаме, че $\top$ е съкратен запис на формулата $\neg\bot$, т. е. на формула, която е винаги вярна.

✓ **2.4.12. Пример.** Нека $=$ и $\in$ са двуместни предикатни символи, а $\mathtt{x}$, $\mathtt{y}$ и $\mathtt{z}$ са променливи. Тогава изразът*

$$\forall \mathtt{x}\, \forall \mathtt{y}\, (\forall \mathtt{z}\, (\mathtt{z} \in \mathtt{x} \Leftrightarrow \mathtt{z} \in \mathtt{y}) \Rightarrow \mathtt{x} = \mathtt{y})$$

е съкратен запис на формулата

$$\forall \mathtt{x}\, \forall \mathtt{y}\, (\forall \mathtt{z}\, ((\in(\mathtt{z}, \mathtt{x}) \Rightarrow \in(\mathtt{z}, \mathtt{y})) \,\&\, (\in(\mathtt{z}, \mathtt{y}) \Rightarrow \in(\mathtt{z}, \mathtt{x}))) \Rightarrow =(\mathtt{x}, \mathtt{y}))$$

✓ **2.4.13.** За улеснение, когато записваме формули може да използваме по-неформален запис. Вече споменахме, че атомарните формули от вида $=(+(0, 0), 0)$ ще бъдат записвани по-четливо като $0 + 0 = 0$. Освен това ще си позволяваме да изпускаме някои от скобите. Например може да считаме, че всяка формула от вида $\varphi_1 \vee \varphi_2 \vee \varphi_3 \vee \varphi_4$ е съкратен запис

*В теория на множествата този израз се нарича аксиома за екстенсионалност или аксиома за обемност.

на формулата $(((\varphi_1 \vee \varphi_2) \vee \varphi_3) \vee \varphi_4)$. Ще считаме, че импликацията ($\Rightarrow$) и еквиваленцията ($\Leftrightarrow$) са с най-малък приоритет, конюнкцията ($\&$) и дизюнкцията ($\vee$) са със среден (и равен по между си) приоритет и унарните операции ($\neg$, $\forall$ и $\exists$) са с най-голям приоритет. Например всяка формула от вида $\varphi_1 \,\&\, \varphi_2 \Rightarrow \varphi_3 \vee \varphi_4$ е съкратен запис на формулата $((\varphi_1 \,\&\, \varphi_2) \Rightarrow (\varphi_3 \vee \varphi_4))$.

Дефиницията на формула е дадена по такъв начин, че да направи вярно следното твърдение:

2.4.14. Твърдение за еднозначен синтактичен разбор. *Всяка формула притежава единствено синтактично дърво.*

От еднозначността на синтактичния разбор следва например, че никоя формула не може да бъде едновременно от вида $(\varphi_1 \& \varphi_2)$ и $(\psi_1 \vee \psi_2)$, нито пък едновременно от вида $(\varphi_1 \Rightarrow \varphi_2)$ и $\forall \mathtt{x}\, \psi$. Също така от тук следва например, че една съждителна формула може да бъде едновременно от вида $(\varphi_1 \vee \varphi_2)$ и $(\psi_1 \vee \psi_2)$ само когато $\varphi_1 = \psi_1$ и $\varphi_2 = \psi_2$.

2.4.15. Пример. Синтактичното дърво на формулата

$$\forall \mathtt{x}\, (\mathtt{p}(\mathtt{x}) \Rightarrow \exists \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r}(\mathtt{x}))$$

е илюстрирано във фигура 11.

✓ ***Задача* 17:** Кое е синтактичното дърво на следната формула:

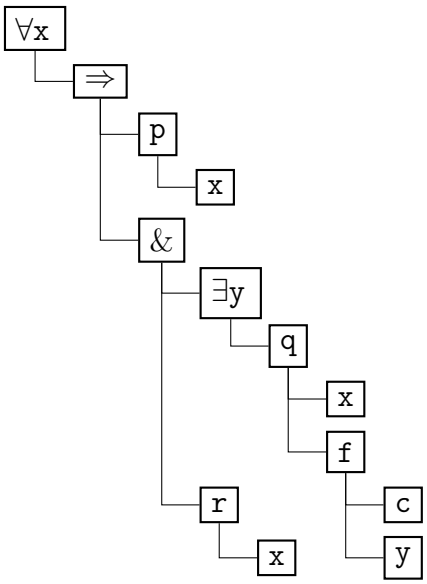
$$\exists \mathtt{x}\, (\forall \mathtt{x}\, \forall \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r}(\mathtt{x}) \Rightarrow \mathtt{p}(\mathtt{x}))$$

Грешка ще бъде да считаме, че всяка формула има фиксиран смисъл. Да разгледаме например следната формула:

$$\mathtt{x} + 0 = \mathtt{x} \tag{5}$$

В тази формула $=$ е двуместен предикатен символ, $+$ е двуместен функционален символ, 0 е символ за константа и $\mathtt{x}$ е променлива. Какво ни казва тази формула? Например какви стойности може да приема променливата $\mathtt{x}$? Естествени числа, множества от думи или квадратни матрици 3×3? Какъв е смисълът на символа $+$? Събиране на естествени числа, обединение на множества или събиране на матрици?

По-нататък ще дефинираме понятието структура. Именно структурата ще дава конкретен смисъл на символите, които използваме във формулите. При различните структури тези символи ще придобиват



Фиг. 11. Синтактично дърво за формулата $\forall \mathtt{x}\,(\mathtt{p}(\mathtt{x}) \Rightarrow \exists \mathtt{y}\,\mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r}(\mathtt{x}))$

различен смисъл и следователно е безсмислено да питаме за една формула дали е вярна, или не, без да сме казали коя е структурата, спрямо която интерпретираме символите във формулата. Тъй като смисълът на формулите зависи от структурата, спрямо която ги интерпретираме, то значи при някои структури една формула може да бъде вярна, а при други — не.

2.4.16. Пример. Да разгледаме няколко различни структури, в които да интерпретираме формулата $\mathtt{x} + 0 = \mathtt{x}$.

а) В първата структура, която ще разгледаме, нека възможните стойности на променливата $\mathtt{x}$ са естествените числа. Нека „=“ е равенство, символът за константа 0 е числото нула, а „+“ е събиране на естествени числа. В тази структура формулата $\mathtt{x}+0 = \mathtt{x}$ ни казва, че всяко естествено число, събрано с нула дава същото естествено число. Следователно в тази структура формулата е вярна.

б) Сега да разгледаме структура, която е същата като предходната, само че символът 0 се интерпретира като числото едно. В тази структура формулата ни казва, че всяко естествено число, събрано с единица, дава същото естествено число. Следователно в тази структура формулата не е вярна.

2.6. Свободни и свързани променливи

Уводни бележки

Променливите в математиката и програмирането са два вида — свободни и свързани. Двата вида променливи са много различни един от друг. Да разгледаме израза

$$\sum_{i=1}^{n} i^2$$

За да пресметнем този израз е нужно да знаем каква е стойността на променливата n, но не и стойността на променливата i. Въпреки че в този израз се срещат две променливи — n и i – очевидно тези променливи са много различни. Ако искаме с горния израз да дефинираме функция, тази функция ще зависи само от n, но не и от i:

$$f(n) = \sum_{i=1}^{n} i^2$$

Променливите, подобни на n, се наричат *свободни променливи*, а подобните на i се наричат *свързани променливи*.

Ето още няколко примера. В следващия израз променливата x е свързана, а y е свободна:

$$\int_0^1 f(x,y)dx$$

В следния програмен блок променливата `i` е свързана, а променливите `a` и `x` са свободни:

```
{
   int i;
   for(i=1;i<1000;i++){
      x[i] = (x[i-1]*x[i-1]) % a;
   }
}
```

И така, да обобщим — стойността (смисъла) на един израз зависи от стойността на свободните променливи, срещащи се в него, но не зависи от стойността на свързаните променливи в израза.

Едно друго различие между свободните и свързаните променливи е следното. Ако в един израз заменим всяко срещане на свободната променлива x с някакъв израз, ще се получи пак смислен израз. Да заменяме свързана променлива с израз не е позволено.

Например ако в израза

$$\sum_{i=1}^{n} i^2$$

заменим n с k^2+5 получаваме израза

$$\sum_{i=1}^{k^2+5} i^2$$

Ако в израза

$$\int_0^1 f(x,y)dx$$

заменим y с z^2+5 получаваме израза

$$\int_0^1 f(x,z^2+5)dx$$

Ако в горния програмен блок заменим променливата `a` с израза `a*a+5` получаваме новия програмен блок:*

```
{
   int i;
   for(i=1;i<1000;i++){
      x[i] = (x[i-1]*x[i-1]) % (a*a+5);
   }
}
```

Ако по подобен начин заменяме свързаните променливи с изрази, обикновено се получават безсмислици. Например ако в израза

$$\sum_{i=1}^{n} i^2$$

заменим i с k^2+5 получаваме

$$\sum_{(k^2+5)=1}^{n} (k^2+5)^2$$

*При подобни замени трябва да се спазват типовете. Например ако променливата `a` се срещаше отляво на оператор за призвояване, тогава бихме могли да заменяме тази променлива само с израз, притежаващ определен адрес в паметта. В този случай замяната на `a` с `a*a+5` не би била коректна, но замяната с `x[4]` ще бъде коректна.

Ако в израза

$$\int_0^1 f(x,y)dx$$

заменим x с z^2+5 получаваме

$$\int_0^1 f(z^2+5,y)d(z^2+5)$$

Ако в горния програмен блок заменим `i` с `a*a+5` получаваме

```
{
   int a*a+5;
   for(a*a+5=1;a*a+5<1000;(a*a+5)++){
      x[i] = (x[i-1]*x[i-1]) % (a*a+5);
   }
}
```

Да забележим, че във всеки от горните случаи свързаната променлива има нещо като „свързващ оператор“, от който съвсем ясно личи, че се получава безсмислица. Това са $\sum_{(k^2+5)=1}^{n}$, $\int_0^1 \ldots d(z^2+5)$ и декларацията `int a*a+5;`.

При преименуване на свързана променлива смисълът на израза се запазва. Например

$$\sum_{i=1}^{n} i^2 = \sum_{j=1}^{n} j^2 \neq \sum_{i=1}^{m} i^2$$

$$\int_0^1 f(x,y)dx = \int_0^1 f(z,y)dz \neq \int_0^1 f(x,z)dx$$

Също и в горния програмен блок смисълът се запазва, ако преименуваме `i` на `j`, но не и ако преименуваме `a` на `b`.

Една друга разлика между свободните и свързаните променливи е това, че свободните променливи имат глобална видимост, а свързаните — локална. Да разгледаме следния програмен фрагмент:

```
{
   int i, j;
   i = a * a;
   {
```

свободни променливи	*свързани променливи*
изразът зависи от стойността им	стойността им не е релевантна
може да се заместват с изрази	не може да се заместват
не може да се преименуват	може да се преименуват
глобална видимост	локална видимост

Таблица 3. Свойства на свободните и свързаните променливи

```
      int i;
      i = b + 5;
      j = 3;
   }
   {
      int i, a;
      i = b + j;
      a = i * i;
      b = a * i;
   }
}
```

В този фрагмент са декларирани три променливи с име `i`. Въпреки че използват едно и също име, тези три променливи са напълно различни една от друга. Освен това всяка си има своя област на видимост. Променливите `i`, декларирани във вътрешните блокове, са видими само в рамките на тези блокове. Променливата `i`, декларирана във външния блок, също е видима само в рамките на този блок, но не и извън него. Освен това вътрешните променливи `i` скриват външната променлива `i`, в следствие на което тя не се вижда във вътрешните блокове. Променливата `j` обаче не се скрива от вътрешна декларация и затова се вижда и във вътрешните блокове.

В този програмен фрагмент се използват и две свободни променливи — `a` и `b`. Всяко срещане на променлива `b` в този програмен фрагмент се отнася за една и съща променлива. Във втория вътрешен блок обаче е декларирана свързана променлива `a`. Тази свързана променлива `a` скрива свободната променлива `a`.

При свързаните променливи може да има различни свързани променливи с едно и също име, но при свободните това е невъзможно — едноименните свободни променливи винаги са една и съща променлива.

Казаното дотук за свободните и свързаните променливи е резюмирано в таблица 3.

– Ако заменим в тази формула променливата $\mathtt{x}$ с терма $\mathtt{f(x)}$ получаваме безсмислица: $\forall \mathtt{f(x)}\, \mathtt{p(f(x))}$.
– Ако преименуваме в тази формула променливата $\mathtt{x}$ с $\mathtt{y}$, получаваме формула с напълно същия смисъл: $\forall \mathtt{y}\, \mathtt{p(y)}$.
– Кванторите създават локална видимост за променливата си. Ако в контекста на формулата $\forall \mathtt{x}\, \mathtt{p(x)}$ се срещат някакви променливи $\mathtt{x}$, то това са напълно различни променливи $\mathtt{x}$. Например първата променлива $\mathtt{x}$ от формулата $\mathtt{q(x)} \,\&\, \forall \mathtt{x}\, \mathtt{p(x)}$ е напълно различна променлива $\mathtt{x}$ (която впрочем е свободна), от втората и третата променлвива $\mathtt{x}$, която е свързана.

Подформули

За да направим по-ясно кои са свързаните и кои са свободните променливи във формула, ще въведем едно помощно понятие:

✓ **2.6.1. Дефиниция.** Когато формулата ψ е част от формулата φ (т.е. ψ е подниз на φ), казваме, че ψ е *подформула* на φ.

✓ **2.6.2.** За да разберем правилно смисъла на току-що дадената дефиниция, е най-добре да си мислим не за самите формули, които са някакви редици от символи, а за техните синтактични дървета. Ако си мислим за синтактичното дърво на една формула, някои от поддърветата ѝ ще бъдат нейни подформули, а други (по-малките) — термове. Подформулите на една формула се получават от всички поддървета, които са формули, а не термове. Областта на действие на всеки квантор е поддървото, което се намира под съответния квантор. Разгледайте фигура 12, в която е илюстрирано синтактичното дърво на формулата

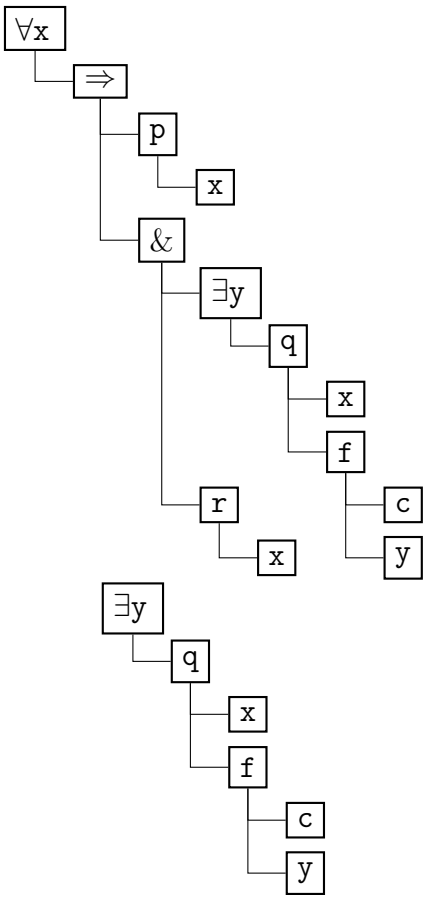
$$\forall \mathtt{x}\, (\mathtt{p(x)} \Rightarrow \exists \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r(x)})$$

както и синтактичното дърво на областта на действие на квантора $\exists \mathtt{y}$ в тази формула.

Ако решим да си мислим не за синтактичните дървета, а за самите формули като редици от символи, тогава дефиниция 2.6.1 ще бъде вярна само тогава, когато изписваме формулата изцяло, без да пропускаме нито едни скоби. Да разгледаме отново формулата

$$\forall \mathtt{x}\, (\mathtt{p(x)} \Rightarrow \exists \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r(x)})$$

Според дефиниция 2.6.1, ако една подформула започва с квантор, тогава тази подформула е областта на действие на този квантор. Една



Фиг. 12. Синтактично дърво на формула и подформула

подформула на горната формула е формулата $\exists \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r}(\mathtt{x})$. Тази формула започва с квантора $\exists \mathtt{y}$ и значи излиза, че тя трябва да е областта на действие на този квантор. Достатъчно е обаче да погледнем дървото от фигура 12, за да видим, че това е невярно!

Тази грешка се обяснява по следния начин — не е вярно, че формулата $\exists \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r}(\mathtt{x})$ започва с квантор. Ако я напишем изцяло, без да пропускаме скоби, тогава ще видим че тази формула започва не с квантор, а със скоба, защото дефиниция 2.4.10 в) изисква около всяка конюнкция да се слагат скоби:

$$(\exists \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r}(\mathtt{x}))$$

Следователно тази подформула започва не с квантор, а със скоба.

Тъй като е много неудобно всеки път да преценяваме къде се слагат всички нужни скоби и дали дадена подформула започва с квантор,

или със скоба, която не е написана, най-добре е когато откриваме подформулите на една формула, да си мислим за синтактичното дърво на формулата, а не за самата формула.

✓ ***Задача* 18:** Намерете всички подформули, на формулата от фигура 12.

Свързани и свободни променливи във формула

Ако приложим всичко казано до момента за свободните и свързаните променливи в една формула, ще получим следната дефиниция:

✓ **2.6.3. Дефиниция.**

а) Когато формулата φ съдържа подформула от вида $\forall \mathtt{x}\, \psi$ или $\exists \mathtt{x}\, \psi$ (където ψ е формула), казваме, че тази подформула е *областта на действие* на квантора $\forall \mathtt{x}$ или $\exists \mathtt{x}$, с който започва подформулата.

б) Едно срещане на променливата $\mathtt{x}$ в някоя формула е *свързано*, ако попада в областта на действие на някой квантор $\forall \mathtt{x}$ или $\exists \mathtt{x}$.

в) Възможно е едно срещане на променливата $\mathtt{x}$ в някоя формула да попада едновременно в областта на действие на няколко квантора $\forall \mathtt{x}$ или $\exists \mathtt{x}$. Измежду всички тези квантори, за квантора, чиято област на действие е най-малка,* казваме, че *управлява* това срещане на променливата $\mathtt{x}$.

г) Едно срещане на променливата $\mathtt{x}$ в някоя формула е *свободно*, ако не попада в областта на действие на никой квантор $\forall \mathtt{x}$ или $\exists \mathtt{x}$.

д) Променливата $\mathtt{x}$ е *свободна променлива* на формулата φ, ако $\mathtt{x}$ се среща в φ на място, което не попада в областта на действие на никой квантор $\forall \mathtt{x}$ или $\exists \mathtt{x}$. С други думи $\mathtt{x}$ е свободна променлива, ако $\mathtt{x}$ има свободно срещане във формулата.

Много е важно да се разберат следващите три примера.

✓ **2.6.4. Пример.** В безкванторните формули няма квантори, следователно не е възможно една променлива да попадне в областта на действие на квантор. Следователно всички променливи, които се срещат в безкванторните формули са техни свободни променливи. Например променливите $\mathtt{x}$, $\mathtt{y}$ и $\mathtt{z}$ са свободните променливи на формулата

$$(\mathtt{p}(\mathtt{x}, \mathtt{c}) \vee \neg \mathtt{q}(\mathtt{y})) \Rightarrow \mathtt{p}(\mathtt{z}, \mathtt{x})$$

*Тъй като всички тези подформули се включват една в друга, е все едно как ще уточним смисъла на думата „най-малка“: най-малка по дължина или най-малка по включване.

Атомарните формули са вид безкванторни формули, следователно всички променливи, които се срещат в атомарните формули са техни свободни променливи.

✓ **2.6.5. Пример.** Променливите $\mathtt{x}$ е единствената свободна променлива на формулата

$$\forall \mathtt{x}\, \forall \mathtt{y}\, (\mathtt{p}(\mathtt{x}, \mathtt{c}) \vee \neg \mathtt{q}(\mathtt{y})) \Rightarrow \forall \mathtt{z}\, \mathtt{p}(\mathtt{z}, \mathtt{x})$$

В тази формула променливата $\mathtt{x}$ от подформулата $\mathtt{p}(\mathtt{x}, \mathtt{c})$ е свързана, защото попада в областта на действие на квантора $\forall \mathtt{x}$. Същевременно променливата $\mathtt{x}$ от подформулата $\mathtt{p}(\mathtt{z}, \mathtt{x})$ е свободна, защото не попада в областта на действие на квантор $\forall \mathtt{x}$ или $\exists \mathtt{x}$.

Също както в езиците за програмиране декларирането на локална променлива в даден програмен блок скрива едноименните променливи, декларирани в по-външен блок, така и кванторите скриват едноименните свободни променливи, както и свързаните променливи с по-външен квантор.

✓ **2.6.6. Пример.** Във формулата

$$\forall \mathtt{x}\, (\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}) \vee \mathtt{p}(\mathtt{x})) \vee \mathtt{p}(\mathtt{x})$$

в първата атомарна формула $\mathtt{p}(\mathtt{x})$ променливата $\mathtt{x}$ се управлява от втория квантор, във втората атомарна формула $\mathtt{p}(\mathtt{x})$ променливата $\mathtt{x}$ се управлява от първия квантор, а в третата — променливата $\mathtt{x}$ е свободна.

✓ ***Задача* 19:** За всяка променлива, срещаща се във формулата

$$\forall \mathtt{x}\, (\forall \mathtt{x}\, \exists \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{f}(\mathtt{c}, \mathtt{y})) \,\&\, \mathtt{r}(\mathtt{x}) \Rightarrow \mathtt{p}(\mathtt{x}, \mathtt{y}))$$

определете дали е свободна или свързана. Кои са кванторите, управляващи свързаните променливи? Коя е областта на действие на всеки един от кванторите? Кое е множеството от свободните променливи на тази формула?

Преименуване на свързаните променливи

В уводните бележки на този раздел бе споменато, че свързаните променливи в един израз може да се преименуват без това да промени смисъла на израза. Ако формулата φ може да се сведе до формулата ψ посредством преименуване на свързаните променливи, казваме, че тези две формули са *конгруентни* (точна дефиниция на това понятие е дадена малко по-долу). Ако две формули са конгруентни, то техният смисъл е един и същ.

2.6.7. Пример. Формулите $\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}, \mathtt{y})$ и $\forall \mathtt{z}\, \mathtt{p}(\mathtt{z}, \mathtt{y})$ са конгруентни. Втората формула може да се получи от първата като преименуваме $\mathtt{x}$ на $\mathtt{z}$. Тези формули обаче не са конгруентн с формулата $\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}, \mathtt{t})$, защото променливата $\mathtt{y}$ е свободна и не може да се преименува.

Ако във формулата $\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}, \mathtt{y})$ преименуваме свързаната променлива $\mathtt{x}$ на $\mathtt{y}$, получаваме формулата $\forall \mathtt{y}\, \mathtt{p}(\mathtt{y}, \mathtt{y})$. Тези две формули обаче не са конгруентни, защото новопоявилият се квантор $\forall \mathtt{y}$ обхваща и свободната променлива $\mathtt{y}$ и я превръща от свободна в свързана.

Можем да дефинираме конгруентните формули по следния начин:

✓ **2.6.8. Дефиниция.** Две формули φ и ψ са *конгруентни*, ако са изпълнени следните условия:

а) двете формули имат една и съща дължина;

б) ако на дадена позиция в едната формула има символ, който не е променлива, то в другата формула стои същият символ;

в) ако на дадена позиция в едната формула има променлива и тя е свободна променлива, то на същата позиция в другата формула стои същата променлива и тя отново е свободна променлива;

г) ако на дадена позиция в едната формула има променлива и тя е свързана променлива, то на същата позиция в другата формула също стои променлива и тя отново е свързана променлива. Управляващите квантори на тези две променливи стоят на една и съща позиция.

✓ ***Задача* 20:** Конгруентни ли са формулите:

а) $\forall \mathtt{x}\, \forall \mathtt{y}\, \forall \mathtt{y}\, \mathtt{p}(\mathtt{x}, \mathtt{y})$ и $\forall \mathtt{x}\, \forall \mathtt{z}\, \forall \mathtt{y}\, \mathtt{p}(\mathtt{x}, \mathtt{y})$

б) $\forall \mathtt{x}\, \forall \mathtt{y}\, \forall \mathtt{y}\, \mathtt{p}(\mathtt{x}, \mathtt{y})$ и $\forall \mathtt{y}\, \forall \mathtt{x}\, \forall \mathtt{x}\, \mathtt{p}(\mathtt{y}, \mathtt{x})$

в) $\forall \mathtt{x}\, \forall \mathtt{y}\, \forall \mathtt{y}\, \mathtt{p}(\mathtt{x}, \mathtt{y})$ и $\forall \mathtt{y}\, \forall \mathtt{y}\, \forall \mathtt{x}\, \mathtt{p}(\mathtt{y}, \mathtt{x})$

След като разгледаме внимателно условията в дефиницията за конгруентност, може да забележим, че тя дефинира рефлексивна, симетрична и транзитивна релация. Следователно конгруентността е релация на еквивалентност и е вярно следното следствие:

✓ **2.6.9. Следствие.**

а) $\varphi \equiv \varphi$

б) ако $\varphi \equiv \psi$, *то* $\psi \equiv \varphi$

в) ако $\varphi \equiv \psi$ *и* $\psi \equiv \chi$, *то* $\varphi \equiv \chi$

Вече споменахме, че ще искаме така да разработваме теорията на предикатната логика, че да можем да отъждествяваме конгруентните формули, т.е. да работим „с точност до конгруентност“. Това означава, че трябва да бъдат верни твърдения, подобни на следващото.

2.6.10. Твърдение. *Ако две формули са конгруентни, то те имат едни и същи свободни променливи.*

Доказателство. Дефиницията на конгруентност ни гарантира, че ако на дадена позиция в едната формула има свободна променлива, то на същата позиция в другата формула стои същата променлива и тя също е свободна. ∎

За да не се усложняваме ненужно, така формулирахме дефиницията за формула (2.4.10), че да можем да пишем „объркващи“ формули като

$$\forall \mathtt{x}\, \forall \mathtt{y}\, (\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}, \mathtt{y}) \vee \mathtt{q}(\mathtt{x}, \mathtt{y}))$$

В тази формула има два квантора с променливата $\mathtt{x}$. Променливата $\mathtt{x}$ в подформулата $\mathtt{p}(\mathtt{x}, \mathtt{y})$ ще бъде управлявана от втория квантор $\forall \mathtt{x}$, а променливата $\mathtt{x}$ в подформулата $\mathtt{q}(\mathtt{x}, \mathtt{y})$ — от първия квантор $\forall \mathtt{x}$. Но въпреки, че използването на такива формули се позволява от дефинициите, обикновено е добре да пишем „нормални“ и по-лесни за осмисляне формули, в които няма квантори с една и съща променлива. Това винаги е възможно, защото ако има квантори с повтарящи се променливи, с подходящо преименуване на свързаните променливи може да получим конгруентна формула, в която няма квантори с една и съща променлива.

Друг вид „обърквща“ формула е например следната:

$$\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}) \Rightarrow \mathtt{q}(\mathtt{x})$$

И тази формула е обърквща, защото в нея променливата $\mathtt{x}$ се среща хем като свободна, хем като свързана. В подформулата $\mathtt{p}(\mathtt{x})$ тя е свързана, а в $\mathtt{q}(\mathtt{x})$ — свободна. Но и тук няма проблем да се освободим от тази странност като преименуваме свързаната променлива $\mathtt{x}$ на някоя друга.

Ако пък Θ е някакво (крайно) множество от променливи, които по някакви причини „не харесване“, то отново — няма да има проблеми така да преименуваме, че да намерим конгруентна формула, в която няма да има квантори с променливи от Θ. Следващата лема доказва всички тези неща накуп.

✓ **2.6.11. Лема за преименуване на свързаните променливи.** *За всяка формула φ и крайно множество Θ от променливи можем да намерим конгруентна на φ формула, в която не се срещат квантори с променливи от Θ, всички квантори са с различни променливи и никоя от променлива не се среща едновременно като свързана и свободна.*

Доказателство. Нека Θ' е обединението на Θ с множеството от всички променливи (свързани и несвързани), които се срещат в φ. Множеството Θ' е крайно.

Ще построим редица от конгруентни формули $\varphi = \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n$, в която последната формула ще отговаря на изискванията на лемата — всички кванторни променливи ще са различни и никоя кванторна променлива няма да е елемент на Θ'.

Ако разполагаме с формулата φ_i, можем да получим формулата φ_{i+1} по следния начин. Нека $\forall \mathtt{z}\, \psi$ или $\exists \mathtt{z}\, \psi$ е произволна подформула на φ_i, чиято променлива $\mathtt{z}$ е елемент на Θ'. Нека $\mathtt{z}'$ е произволна променлива, която не се среща нито в φ_i, нито е елемент на Θ'. Да преименуваме всички управлявани от квантора $\forall \mathtt{z}$ или $\exists \mathtt{z}$ променливи (в това число и самата кванторна променлива) от $\mathtt{z}$ на $\mathtt{z}'$. Нека така получената конгруентна формула бъде φ_{i+1}.

На всяка стъпка броят на кванторите с променливи от Θ' намалява и значи процесът на строене на тази редица от конгруентни формули спира. Това ще се случи тогава, когато стигнем до формула φ_n, в която няма повече квантори с променливи от Θ'. Тъй като включихме в Θ' всички първоначални променливи на формулата, то значи всяка кванторна променлива е била преименувана. При това преименуване обаче винаги избирахме и затова φ_n не съдържа два квантора с една и съща променлива, нито пък променлива, която е едновременно свързана и свободна. ∎

✓ **2.6.12. Пример.** Да видим с конкретен пример как работи току-що доказаната лема. Нека

$$\varphi = \forall \mathtt{y}\, \forall \mathtt{x}\, (\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}, \mathtt{y}) \vee \mathtt{p}(\mathtt{x}, \mathtt{y})) \vee \mathtt{p}(\mathtt{y}, \mathtt{z})$$

и $\Theta = \{\mathtt{y}, \mathtt{z}\}$. В тази формула има много „дефекти“ — квантори с една и съща променлива ($\mathtt{x}$), променлива, която е хем свързана, хем свободна ($\mathtt{y}$), както и квантори с променливи от Θ. Нека поправим тези дефекти по начина, описан в доказателството на лемата. Нека най-напред изберем втория квантор $\forall \mathtt{x}$. Ще заменим променливата му $\mathtt{x}$ с $\mathtt{z}_{13}$ (избирама коя да е променлива, която не се среща нито във формулата,

нито е елемент на Θ). След замяната получаваме

$$\forall \mathtt{y}\, \forall \mathtt{x}\, (\forall \mathtt{z}_{13}\, \mathtt{p}(\mathtt{z}_{13}, \mathtt{y}) \vee \mathtt{p}(\mathtt{x}, \mathtt{y})) \vee \mathtt{p}(\mathtt{y}, \mathtt{z})$$

Сега да изберем другия квантор $\forall \mathtt{x}$. Ще заменим $\mathtt{x}$ например с $\mathtt{y}_{42}$. Получаваме

$$\forall \mathtt{y}\, \forall \mathtt{y}_{42}\, (\forall \mathtt{z}_{13}\, \mathtt{p}(\mathtt{z}_{13}, \mathtt{y}) \vee \mathtt{p}(\mathtt{y}_{42}, \mathtt{y})) \vee \mathtt{p}(\mathtt{y}, \mathtt{z})$$

Накрая да изберем $\forall \mathtt{y}$ и да заменим $\mathtt{y}$ с $\mathtt{z}'''$. Получаваме

$$\forall \mathtt{z}'''\, \forall \mathtt{y}_{42}\, (\forall \mathtt{z}_{13}\, \mathtt{p}(\mathtt{z}_{13}, \mathtt{z}''') \vee \mathtt{p}(\mathtt{y}_{42}, \mathtt{z}''')) \vee \mathtt{p}(\mathtt{y}, \mathtt{z})$$

Така получената формула нито съдържа квантори с една и съща променлива, нито има променлива, която е хем свързана, хем свободна, нито има квантори с променливи от Θ. Същевременно тази формула е конгруентна с φ, защото се получава с преименувания на свързаните променливи.

✓ ***Задача* 21:** Намерете формула, конгруентна на

$$\forall \mathtt{x}\, \forall \mathtt{y}\, (\exists \mathtt{x}\, \mathtt{p}(\mathtt{x}, \mathtt{y}) \,\&\, \mathtt{q}(\mathtt{x}, \mathtt{z}) \Rightarrow \exists \mathtt{z}\, \mathtt{p}(\mathtt{y}, \mathtt{z}))$$

която не съдържа квантори с повтарящи се променливи, никоя променлива не се среща едновременно като свързана и свободна и променливите $\mathtt{x}$ и $\mathtt{y}$ не се срещат.

2.6.13. Лема. *а) Ако* $\forall \mathtt{x}\, \varphi \equiv \forall \mathtt{x}\, \psi$, *то* $\varphi \equiv \psi$.

б) Ако $\exists \mathtt{x}\, \varphi \equiv \exists \mathtt{x}\, \psi$, *то* $\varphi \equiv \psi$.

Доказателство. Двете условия на тази лема се доказват по един и същ начин. Ще докажем първото.

Щом $\forall \mathtt{x}\, \varphi$ и $\forall \mathtt{x}\, \psi$ са с една и съща дължина, то значи и φ и ψ са с една и съща дължина. Щом символите, които не са променливи, в $\forall \mathtt{x}\, \varphi$ и $\forall \mathtt{x}\, \psi$ са едни и същи, то също и в φ и ψ тези символи са едни и същи.

Ако $\mathtt{z}$ е свободна променлива в φ, то това срещане на $\mathtt{z}$ или е свободна и в $\forall \mathtt{x}\, \varphi$, или $\mathtt{z} = \mathtt{x}$, в който случай началният квантор $\mathtt{x}$ ще управлява $\mathtt{z}$. В първия случай от конгруентността на формулите $\forall \mathtt{x}\, \varphi$ и $\forall \mathtt{x}\, \psi$ получаваме, че на същата позиция във втората формула също стои променливата $\mathtt{z}$ и тя е свободна. Щом това срещане на $\mathtt{z}$ е свободно в $\forall \mathtt{x}\, \psi$, то същото срещане ще е свободно и в ψ. Във втория случай началният квантор $\mathtt{x}$ управлява $\mathtt{z}$. Тъй като формулите $\forall \mathtt{x}\, \varphi$ и $\forall \mathtt{x}\, \psi$ са конгруентни, то във втората променлива на позицията на $\mathtt{z}$ също стои

как можем да заменяме свободните променливи с други термове. Найнапред да дефинираме понятието, което ще бъде играе главната роля в този раздел:

✓ **2.7.1. Дефиниция.** *Субституцията* е функция, която на всяка променлива съпоставя терм.

Забележка: Тъй като субституциите се използват в различни алгоритми, дефиницията им обикновено включва допълнителното изискване, че само за краен брой променливи $s(\mathtt{x}) \neq \mathtt{x}$, а за всички останали $s(\mathtt{x}) = \mathtt{x}$. По този начин, за да представим в компютъра една субституция, е достатъчно да помним само двойките $\langle \mathtt{x}, s(\mathtt{x})\rangle$, при които $\mathtt{x} \neq s(\mathtt{x})$, а те са само краен брой. Това допълнително условие усложнява ненужно разсъжденията и затова тук го изпускаме. Въпреки това може да забележим, че субституциите, които се появяват в разглежданите в този курс алгоритми, всъщност удовлетворяват това допълнително изискване.

Тъй като в термовете и безкванторните формули няма свързани променливи, прилагането на субституция към такива изрази става по възможно най-простия и естествен начин — ако s е субституция, просто трябва да заменим в израза всяка променлива $\mathtt{x}$ с $s(\mathtt{x})$:

✓ **2.7.2. Дефиниция.** Ако s е субституция, а τ — терм или предикатна формула без квантори, то *резултатът от прилагането на субституцията s към τ* се получава като заместим в τ всяко срещане на променливата $\mathtt{x}$ с терма $s(\mathtt{x})$.

2.7.3. Означение. За резултата от прилагане на субституция s към израз τ най-често се използва означението τs. Например ако $\mathtt{x}$ е променлива, то $\mathtt{x}s = s(\mathtt{x})$.

2.7.4. Твърдение. *Нека s е субституция.*

а) Ако τ е терм, то τs е терм.

б) Ако φ е безкванторна формула, то φs е безкванторна формула.

Доказателство. Твърдението може да се докаже директно с помощта на индукция. Освен това то следва и от по-общата лема 2.7.5, която ще използваме и по-нататък. ∎

2.7.5. Лема. *а) Ако τ е терм и изразът τ' се получава като по напълно произволен начин заменим в τ някои променливи с термове, то τ' е терм.*

2.7.7. Пример. Нека $s = [\mathtt{x}, \mathtt{y} \coloneqq \mathtt{f}(\mathtt{y}), \mathtt{x}]$ и $\tau = \mathtt{g}(\mathtt{x}, \mathtt{y}, \mathtt{f}(\mathtt{x}))$. Тогава $\tau s = \mathtt{g}(\mathtt{f}(\mathtt{y}), \mathtt{x}, \mathtt{f}(\mathtt{f}(\mathtt{y})))$.

✓ ***Задача* 25:** Намерете

$$\mathtt{f}(\mathtt{x}, \mathtt{y})[\mathtt{x}, \mathtt{y} \coloneqq \mathtt{y}, \mathtt{y}]$$
$$\mathtt{f}(\mathtt{x}, \mathtt{y})[\mathtt{x}, \mathtt{y} \coloneqq \mathtt{g}(\mathtt{x}), \mathtt{x}]$$
$$\mathtt{f}(\mathtt{g}(\mathtt{x}), \mathtt{y})[\mathtt{x}, \mathtt{y} \coloneqq \mathtt{g}(\mathtt{x}), \mathtt{x}]$$
$$\mathtt{f}(\mathtt{x}, \mathtt{g}(\mathtt{y}))[\mathtt{x}, \mathtt{y} \coloneqq \mathtt{g}(\mathtt{x}), \mathtt{x}]$$

Прилагане на субституция в общия случай

> *Повечето формулировки на правилото за субституция, които са били публикувани — дори и от най-способните логици — преди 1940, са явно сбъркани.*
>
> Хаскел Къри *и* Робер Фейс *[8]*

Остава да видим как можем да прилагаме субституция към произволна предикатна формула. При термовете и безкванторните формули беше лесно. Например за произволна безкванторна формула φ ако s е субституция, то формулата φs се получава като заменим в φ всяка променлива $\mathtt{x}$ с $s(\mathtt{x})$. Можехме да използваме такава проста дефиниция, защото в термовете, атомарните формули и безкванторните формули няма свързани променливи.

Да видим защо при наличието на свързани променливи прилагането на субституции трябва да се прави по-внимателно. Това е така, защото при прилагането на субституция към израз със свързани променливи може да се допуснат два вида грешки.

✓ **2.7.8. Първия вид грешка** вече я споменахме — при прилагане на субституция не трябва да заменяме свързаните променливи, защото се получават безсмислици. Например ако в израза

$$\sum_{i=1}^{100} i^2$$

заменим i с x^2, ще получим безсмислицата

$$\sum_{x^2=1}^{100} (x^2)^2$$

и ако в предикатната формула

$$\forall \mathtt{x}\, \mathtt{p}(\mathtt{x})$$

заменим $\mathtt{x}$ с терма $\mathtt{f}(\mathtt{y})$, ще получим безсмислицата

$$\forall(\mathtt{f}(\mathtt{y}))\, \mathtt{p}(\mathtt{f}(\mathtt{y}))$$

Извод: Когато прилагаме субституция, трябва да игнорираме променливите, които са свързани от някой квантор. Например ако към формулата

$$\mathtt{p}(\mathtt{x}) \vee \forall \mathtt{x}\, \mathtt{q}(\mathtt{x})$$

искаме да приложим субституция, заменяща променливата $\mathtt{x}$ с терма $\mathtt{f}(\mathtt{y})$, ще получим формулата

$$\mathtt{p}(\mathtt{f}(\mathtt{y})) \vee \forall \mathtt{x}\, \mathtt{q}(\mathtt{x})$$

✓ **2.7.9. Вторият вид грешка** е по-лесно да остане незабелязан и затова трябва да сме внимателни. Да предположим, че сме дефинирали числовата функция f така:

$$f(y) = \int_0^1 (x^2 + y^3)\, dx$$

Това, че в дефиницията на тази функция сме използвали променливата x не означава, че от тук нататък до края на света нямаме право да използваме променливи x за други цели. Да допуснем, че в даден момент сме дефинирали някаква променлива x. На колко ще бъде равно $f(2x)$? Ако просто навсякъде заменим y с $2x$ и кажем, че

$$f(2x) = \int_0^1 (x^2 + (2x)^3)\, dx$$

това ще бъде грешка! Да забележим, че изразът от дясната страна на равенството по никакъв начин не зависи от стойността на x, защото променливата x е свързана. Верният отговор е следният:

$$f(2x) = \int_0^1 (z^2 + (2x)^3)\, dz$$

Разбира се, вместо z можем да използваме коя да е друга променлива, която не се използва за други цели.

Ето друг пример. Нека

$$a_n = \sum_{i=1}^{n} i^3$$

Ако след време дефинираме променлива i, на колко ще бъде равно a_{2i}? Верният отговор е

$$a_{2i} = \sum_{j=1}^{2i} j^3$$

Тук също, вместо j можем да използваме коя да е друга променлива, която не се използва за други цели.

Извод: Когато прилагаме субституция, трябва да преименуваме кванторните променливи, ако има опасност някой квантор да „свърже“ променлива, която ще се постави от субституцията и трябва да остане свободна. Например ако към формулата

$$\forall \mathtt{y}\, \mathtt{q}(\mathtt{x}, \mathtt{y})$$

искаме да приложим субституция, заменяща променливата $\mathtt{x}$ с терма $\mathtt{f}(\mathtt{y})$, ще трябва най-напред да преименуваме свързаната променлива $\mathtt{y}$ например на $\mathtt{z}$

$$\forall \mathtt{z}\, \mathtt{q}(\mathtt{x}, \mathtt{z})$$

и едва след това ще може да заменим $\mathtt{x}$ с $\mathtt{f}(\mathtt{y})$:

$$\forall \mathtt{z}\, \mathtt{q}(\mathtt{f}(\mathtt{y}), \mathtt{z})$$

Обмислете добре тези примери, преди да продължите четенето на този раздел! Опитайте се да решите следващата задача и проверете дали отговорът ви е бил верен.

✓ ***Задача* 26:** Още не сме дали точната дефиниция на φs при произволна субституция s и формула φ. Въпреки това, имайки предвид съображенията, изказани дотук, преценете какви трябва да бъдат

а) $(\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}, \mathtt{y}) \vee \mathtt{q}(\mathtt{x}))[\mathtt{x} \coloneqq \mathtt{f}(\mathtt{y})]$

б) $(\forall \mathtt{y}\, \mathtt{p}(\mathtt{x}, \mathtt{y}) \vee \mathtt{q}(\mathtt{y}))[\mathtt{x} \coloneqq \mathtt{f}(\mathtt{y})]$

И така, анализът на грешките от първи и втори тип (вж. 2.7.9 и 2.7.9) показва, че когато субституцията замества едни променливи с термове, съдържащи други променливи, тези променливи — както

заменените, така и новопоявилите се — не трябва да имат нищо общо с кванторните променливи. Ако една заменена променлива е била в областта на действие на квантор, получаваме грешка от първи тип, а когато новопоявила се променлива попадне в областта на действие на квантор, получаваме грешка от втори тип.

За да избегнем грешките от първи тип, в дефиницията трябва да кажем, че заменяме само онези срещания на дадена променлива, които са свободни — в никакъв случай не трябва да променяме свързаните променливи. За да избегнем грешките от втори тип пък, ще трябва да видим кои ще са новите променливи, които ще се появят след като приложим субституцията, и да се погрижим никой от кванторите да не „свърже“ тези нови променливи.

Променливите, които заменя субституцията, са свободните променливи. Ако $\mathtt{z}$ е свободна променлива, тогава субституцията s ще я замени с терма $s(\mathtt{z})$. Следователно новите променливи, които ще се появят след като приложим субституцията, са всички променливи, които се срещат в термовете $s(\mathtt{z})$, където $\mathtt{z}$ е свободна променлива във формулата. Ще наречем такива променливи „опасни“. Изключение е следният очевидно „безопасен“ случай: когато $s(\mathtt{z}) = \mathtt{z}$, тогава заменяме $\mathtt{z}$ с $\mathtt{z}$, т. е. нищо не заменяме и значи не се случва нищо опасно.

✓ **2.7.10. Дефиниция.** Ще наричаме *опасни променливи* за формулата φ при субституция s всички променливи, които се срещат в термовете $s(\mathtt{z})$, където $\mathtt{z}$ е свободна променлива на φ и $s(\mathtt{z}) \neq \mathtt{z}$.

Да забележим, че опасните променливи винаги са краен брой. В предния раздел видяхме, че свързаните променливи могат да се преименуват без това да промени смисъла на формулата. В частност доказахме лема 2.6.11, според която ако си изберем Θ да бъде множеството от опасните променливи, тогава можем да намерим конгруентна формула, която не съдържа нито един квантор, чиято променлива е от това множество. Именно това ни дава възможност да дефинираме прилагането на субституция към произволна формула. Ако във формулата няма квантори с опасни променливи, то няма проблеми да приложим субституцията към всички свободни променливи. В противен случай най-напред намираме конгруентна формула, която не съдържа квантори с опасни променливи и след това прилагаме субституцията към новата формула. Следващата дефиниция формализира всичко това.

✓ **2.7.11. Дефиниция.** Нека s и φ са произволни субституция и формула, а Θ е крайното множество от всички опасни за φ променливи при субституция s. Ако φ не съдържа квантори с променливи от Θ,

нека $\varphi' = \varphi$, а в противен случай нека φ' е конгруентна с φ формула, която не съдържа квантори с променливи от Θ (такава съществува благодарение на лема 2.6.11).

В такъв случай, с φs ще означим израза, който се получава като заместим във φ' всяка срещане на свободна променлива $\mathtt{z}$ с терма $s(\mathtt{z})$. Изразът φs се нарича *резултат от прилагането на субституцията s към φ*.

2.7.12. Да забележим, че според тази дефиниция заменяме само свободните променливи. Ако някое срещане на променливата $\mathtt{z}$ не е свободно, тогава на това място $\mathtt{z}$ не се заменя с терма $s(\mathtt{z})$. В дефиниция 2.7.2 нямаше нужда да правим специална уговорка, че прилагаме субституцията само към свободни променливи, защото в термовете и безкванторните формули всички променливи са свободни. Това впрочем показва, че току-що дадената дефиниция не противоречи на дефиниция 2.7.2, а само я допълва — да заменим всяка променлива $\mathtt{z}$ в терм или безкванторна формула с $s(\mathtt{z})$ е същото каквото да заменим всяка свободна променлива $\mathtt{z}$ с $s(\mathtt{z})$.

✓ **2.7.13. Пример.** Нека s е субституция, за която

$$\begin{aligned} s(\mathtt{x}_1) &= \mathtt{f}(\mathtt{x}_1 + \mathtt{x}_2, \mathtt{z}) \\ s(\mathtt{x}_2) &= \mathtt{y} + \mathtt{x}_3 \\ s(\mathtt{x}_3) &= \mathtt{y} \\ s(\mathtt{y}) &= \mathtt{f}(\mathtt{y}, \mathtt{y}) \\ s(\mathtt{z}) &= (\mathtt{t} + \mathtt{x}_3) + \mathtt{x}_1 \\ s(\mathtt{t}) &= \mathtt{x}_1 + \mathtt{f}(\mathtt{x}_2, \mathtt{t}) \end{aligned}$$

Трябва да приложим тази субституция към формулата

$$\forall \mathtt{x}_3(\mathtt{p}(\mathtt{x}_3, \mathtt{y}) \Rightarrow \exists \mathtt{y}\,(\mathtt{f}(\mathtt{y}, \mathtt{x}_3) = \mathtt{x}_1 + \mathtt{t}))$$

Тъй като тази формула съдържа квантори, трябва да проверим дали някой от кванторите не е с опасна променлива. Свободни променливи са: $\mathtt{y}, \mathtt{x}_1, \mathtt{t}$. Следователно опасни променливи са променливите, срещащи се в термовете $s(\mathtt{y}), s(\mathtt{x}_1), s(\mathtt{t})$, т.е. променливите $\mathtt{y}, \mathtt{x}_1, \mathtt{x}_2, \mathtt{z}, \mathtt{t}$. Кванторът $\exists \mathtt{y}$ е с опасна променлива и трябва да го преименуваме. Например ако преименуваме неговата променлива $\mathtt{y}$ на $\mathtt{y}_{178}$ ще получим

$$\forall \mathtt{x}_3(\mathtt{p}(\mathtt{x}_3, \mathtt{y}) \Rightarrow \exists \mathtt{y}_{178}(\mathtt{f}(\mathtt{y}_{178}, \mathtt{x}_3) = \mathtt{x}_1 + \mathtt{t}))$$

Към тази формула вече може да прилагаме субституцията s (заменяме само свободните променливи). Получаваме

$$\forall \mathtt{x}_3(\mathtt{p}(\mathtt{x}_3, \mathtt{f}(\mathtt{y}, \mathtt{y})) \Rightarrow \exists \mathtt{y}_{178}(\mathtt{f}(\mathtt{y}_{178}, \mathtt{x}_3) = \mathtt{f}(\mathtt{x}_1 + \mathtt{x}_2, \mathtt{z}) + (\mathtt{x}_1 + \mathtt{f}(\mathtt{x}_2, \mathtt{t}))))$$

✓ ***Задача* 27:** Нека s е същата субституция като в пример 2.7.13. Намерете резултатите от прилагането на субституцията s към следните формули:

$$\mathtt{p}(\mathtt{x}_1, \mathtt{z}) \vee \mathtt{x}_1 + \mathtt{x}_2 = \mathtt{f}(\mathtt{t}, \mathtt{y})$$
$$\forall \mathtt{x}_3(\mathtt{p}(\mathtt{x}_3, \mathtt{t}) \Rightarrow \exists \mathtt{y}\,(\mathtt{f}(\mathtt{y}, \mathtt{x}_3) = \mathtt{x}_1 + \mathtt{t}))$$
$$\forall \mathtt{x}_3(\mathtt{p}(\mathtt{x}_3, \mathtt{z}) \Rightarrow \exists \mathtt{y}\,(\mathtt{f}(\mathtt{y}, \mathtt{x}_3) = \mathtt{x}_1 + \mathtt{z}))$$
$$\forall \mathtt{x}_3(\mathtt{p}(\mathtt{x}_3, \mathtt{z}) \Rightarrow \exists \mathtt{y}\,(\mathtt{f}(\mathtt{y}, \mathtt{x}_3) = \mathtt{x}_2 + \mathtt{z}))$$

Свойства на субституциите

> *Това едва ли може да се нарече интуитивна дефиниция за субституция. Тя обаче има точните свойства.*
>
> Джон Рейнолдс *[15]*

В дефиниция 2.7.11 нарекохме φs просто „израз“, а не формула. Всъщност φs е формула, но това се нуждае от доказателство.

2.7.14. Твърдение. *За всяка субституция s и формула φ изразът φs е формула.*

Доказателство. Ако във формулата φ няма квантори с опасни променливи при субституция s, тогава φs се получава просто като заменяме всяко срещане на свободна променлива $\mathtt{z}$ с $s(\mathtt{z})$. Затова искането следва от лема 2.7.5.

Ако във формулата има квантори с опасни променливи, то съгласно дефиниция 2.7.11 трябва най-напред да преименуваме свързаните променливи по подходящ начин, получавайки конгруентна формула φ', след което действаме по вече описания начин. Затова в този случай φs също е формула. ■

Вече споменахме, че ще искаме да отъждествяваме конгруентните формули. Това би създало проблем, ако се окаже, че като приложим някоя субституция към две конгруентни формули, е възможно да получим две съществено различни формули. Следващото твърдение показва, че това за щастие не се случва.

✓ **2.7.15. Твърдение.** *Ако $\varphi \equiv \psi$, то $\varphi s \equiv \psi s$. С други думи, ако формулите φ и ψ са конгруентни, то за произволна субституция s, формулите φs и ψs са конгруентни.*

✓ **2.7.16. Твърдение.** *Нека субституциите s_1 и s_2 съвпадат за всички свободни променливи на формулата φ. Тогава $\varphi s_1 \equiv \varphi s_2$.*

Доказателство. Нека φ' е конгруентна с φ формула, която не съдържа квантори с променливи, които са опасни при субституция s_1. Формулата φ' има същите свободни променливи като φ, а за всяка такава свободна променлива субституциите s_1 и s_2 съвпадат. Това означава, че формулата φ' не съдържа квантори с опасни променливи също и при субституция s_2. От дефиницията за прилагане на субституция (дефиниция 2.7.11) следва, че $\varphi' s_1 = \varphi' s_2$.

От тук получаваме исканото, защото от една страна $\varphi s_1 \equiv \varphi' s_1$, а от друга $\varphi' s_2 \equiv \varphi s_2$. ∎

2.7.17. Твърдение. *Ако са дадени две субституции s_1 и s_2, да дефинираме s да бъде субституцията, за която $s(\mathtt{z}) = (\mathtt{z}s_1)s_2$ за всяка променлива $\mathtt{z}$. Тогава за всяка формула φ*

$$\varphi s \equiv (\varphi s_1)s_2$$

Доказателство. Нека s е субституцията, за която $s(\mathtt{z}) = (\mathtt{z}s_1)s_2$ за всяка променлива $\mathtt{z}$. Ще докажем, че за произволна формула $\varphi s \equiv (\varphi s_1)s_2$.

Нека Θ е множеството от свободните променливи на φ. Нека Θ_1 е обединението на Θ с множеството от всички променливи, които се срещат в термовете $s_1(\mathtt{z})$, където $\mathtt{z} \in \Theta$. В такъв случай Θ_1 ще включва всички опасни променливи в φ при субституция s_1. Нека Θ_2 е обединението на Θ_1 с множеството от всички променливи, които се срещат в термовете $s_2(\mathtt{z})$, където $\mathtt{z} \in \Theta_1$. Множеството Θ_2 е крайно, значи от лемата за преименуване на свързаните променливи 2.6.11 можем да намерим формула ψ, която е конгруентна с φ, и която не съдържа квантори с променливи от Θ_2.

Тъй като Θ_1 съдържа опасните променливи за φ при субституция s_1 и $\Theta_1 \subseteq \Theta_2$, то ψ не съдържа квантори с променливи, които са опасни при субституция s_1. Следователно ψs_1 се получава като заменим в ψ всяка свободна променлива $\mathtt{x}$ с терма $s_1(\mathtt{x})$. Това означава, че свободните променливи на ψs_1 са елементи на Θ_1, а формулата ψs_1 не съдържа квантори с променливи, които са опасни при субституция s_2, следователно $(\psi s_1)s_2$ се получава от ψs_1 като заменим всяка свободна променлива $\mathtt{y}$ с терма $s_2(\mathtt{y})$. Тъй като всички свободни променливи в ψs_1 се съдържат в термовете $s_1(\mathtt{x})$, които са заменили свободните променливи $\mathtt{x}$ в ψ, то това означава, че $(\psi s_1)s_2$ може да се получи от ψ като заменим всяка свободна променлива $\mathtt{x}$ в ψ с терма $(\mathtt{x}s_1)s_1$.

определят неговия носител. Това е така, защото този сорт ще има един и същи смисъл във всички структури.*

3.1.1. Означение.

а) Множеството от всички неща от сорт **индив** в структура $\mathbf{M}$ се означава с $[\![\textbf{индив}]\!]^{\mathbf{M}}$ или с $|\mathbf{M}|$ и се нарича *универсум* или *носител* на структурата.**

б) *Смисълът* или *интерпретацията* в дадена структура $\mathbf{M}$ на символите за константи, функционалните символи и предикатните символи ще означаваме, използвайки семантичните скоби $[\![\cdot]\!]^{\mathbf{M}}$ или посредством горен индекс $^{\mathbf{M}}$. Например:

- $[\![\mathtt{c}]\!]^{\mathbf{M}}$ или $\mathtt{c}^{\mathbf{M}}$ е интерпретацията на символа за константа $\mathtt{c}$ в структурата $\mathbf{M}$,
- $[\![\mathtt{f}]\!]^{\mathbf{M}}$ или $\mathtt{f}^{\mathbf{M}}$ е интерпретацията на функционалния символ $\mathtt{f}$ в структурата $\mathbf{M}$ и
- $[\![\mathtt{p}]\!]^{\mathbf{M}}$ или $\mathtt{p}^{\mathbf{M}}$ е интерпретацията на предикатния символ $\mathtt{p}$ в структурата $\mathbf{M}$.

По отношение на това какъв точно може да бъде универсумът на една структура и как трябва да се интерпретират различните символи в нея, да обърнем внимание на следното.

Универсумът е множество. Не се налага да ограничаваме по какъвто и да е начин какво точно съдържа това множесто — елементите му могат да бъдат числа, функции, други множества и т. н. Единственото ограничение, което се налага да приемем, е това универсумът да бъде непразно множество. Причината, поради която налагаме това ограничение, е следната — когато универсумът е празното множество, се появяват някои странности, които неужно ще усложняват формулировките на твърденията и техните доказателства. И тъй като структурите с празен универсум очевидно не могат да бъдат кой знае колко полезни, струва си да си спестим тези усложнения като забраним структурите с празен универсум.

*Понякога обаче се оказват полезни и структури с нестандартен носител за сорта **съжд**.

**Някои математици използват за структурите удебелени латински букви — $\mathbf{A}$, $\mathbf{B}$, $\mathbf{M}$, $\mathbf{N}$, други калиграфски букви — $\mathcal{A}$, $\mathcal{B}$, $\mathcal{M}$, $\mathcal{N}$, трети готически — $\mathfrak{A}$, $\mathfrak{B}$, $\mathfrak{M}$, $\mathfrak{N}$. Универсумът на структурите се означава или с вертикални черти — $|\mathbf{A}|$, $|\mathbf{B}|$, $|\mathbf{M}|$, $|\mathbf{N}|$, или посредством съответните обикновени латински букви — A, B, M, N.

възможност да забравим, че интерпретираме предикатните символи посредством релации, и да работим с тях така, все едно че ги интерпретираме посредством предикати:

3.1.4. Означение. Ако R е множество от n-торки, то ще използваме изрази от вида

$$R(x_1, x_2, \ldots, x_n)$$

като съкратен запис на

$$\langle x_1, x_2, \ldots, x_n \rangle \in R$$

Благодарение на това означение за всеки предикатен символ $\mathtt{p}$ може да използваме записа $\mathtt{p}^{\mathbf{M}}(x_1, x_2, \ldots, x_n)$, т. е. да го използваме така, все едно че $\mathtt{p}^{\mathbf{M}}$ е функция. В действителност $\mathtt{p}^{\mathbf{M}}(x_1, x_2, \ldots, x_n)$ е съкратен запис на $\langle x_1, x_2, \ldots, x_n \rangle \in \mathtt{p}^{\mathbf{M}}$, но обикновено не се налага да помним това.

Вече сме готови да дадем точната дефиниция на структура за сигнатурата $\mathfrak{sig} = \langle \mathtt{d}_1, \mathtt{d}_2, \mathtt{d}_3, \ldots \rangle$. Това ще бъде редица, в която първият елемент е универсумът на структурата (т. е. множеството от индивидите от сорт **индив**), а останалите елементи са интерпретациите на символите $\mathtt{d}_1, \mathtt{d}_2, \mathtt{d}_3, \ldots$.

✓ **3.1.5. Дефиниция.** Нека $\mathfrak{sig} = \langle \mathtt{d}_1, \mathtt{d}_2, \mathtt{d}_3, \ldots \rangle$ е сигнатура. Редицата $\mathbf{M} = \langle [\![\textbf{индив}]\!]^{\mathbf{M}}, [\![\mathtt{d}_1]\!]^{\mathbf{M}}, [\![\mathtt{d}_2]\!]^{\mathbf{M}}, [\![\mathtt{d}_3]\!]^{\mathbf{M}}, \ldots \rangle$ е *структура* за $\mathfrak{sig}$, ако $[\![\textbf{индив}]\!]^{\mathbf{M}}$ е непразно множество, наречено *универсум* или *носител* на структурата. Освен това:

а) Ако $\mathtt{d}_i$ е символ за константа, то интерпретацията $[\![\mathtt{d}_i]\!]^{\mathbf{M}}$ на $\mathtt{d}_i$ е елемент на универсума.

б) Ако $\mathtt{d}$ е n-местен функционален символ, то интерпретацията $[\![\mathtt{d}_i]\!]^{\mathbf{M}}$ на $\mathtt{d}_i$ е n-местна функция в универсума.

в) Ако $\mathtt{d}$ е n-местен предикатен символ, то интерпретацията $[\![\mathtt{d}_i]\!]^{\mathbf{M}}$ на $\mathtt{d}_i$ е n-местна релация в универсума.

Универсумът на структурата $\mathbf{M}$ се означава с $|\mathbf{M}|$ или $[\![\textbf{индив}]\!]^{\mathbf{M}}$. Интерпретацията на символа $\mathtt{d}$ се означава с $\mathtt{d}^{\mathbf{M}}$ или $[\![\mathtt{d}]\!]^{\mathbf{M}}$.

Интерпретация на символите за константи и функционалните символи в структура:

$$\mathtt{c}^{\mathbf{M}} \in |\mathbf{M}|$$
$$\mathtt{f}^{\mathbf{M}} \colon |\mathbf{M}|^n \to |\mathbf{M}|$$

Интерпретация на предикатните символи:

$$\mathtt{p}^{\mathbf{M}} \colon |\mathbf{M}|^n \to \text{ съждение} \qquad \text{(с предикат)}$$
$$\mathtt{p}^{\mathbf{M}} \subseteq |\mathbf{M}|^n \qquad \text{(с релация)}$$

3.1.6. Дефиниция. а) Когато сигнатурата съдържа двуместен предикатен символ $=$, а структурата интерпретира този символ като равенство, то за тази структура казваме, че е *структура с равенство*.

б) Когато сигнатурата е алгебрична (т.е. единственият предикатен символ в нея е $=$) и структурата интерпретира този символ като равенство, тогава такава структура се нарича *алгебрична структура* или просто *алгебра*. Причината за тази терминология е това, че повечето от структурите, които се използват в алгебрата, са точно такива.*

3.1.7. Пример. Магмите, полугрупите, моноидите, групите, квазигрупите, полупръстените, пръстените, пръстените на Ли, полетата, алгебрите на Клини, полурешетките, решетките и булевите алгебри са само някои от многото примери за алгебрични структури. Например пръстенът на целите числа е структура, чийто универсум е множеството на целите числа, има два символа за константи 0 и 1, които се интерпретират стандартно като числото нула и числото едно, един едноместен функционален символ „$-$“, който се интерпретира като операцията „смяна на знака“, два двуместни функционални символа „$+$“ и „$.$“, които се интерпретират като операциите събиране и умножение и един предикатен символ $=$, който е двуместен и се интерпретира като равенство.

3.1.8. Пример. Графите са пример за структури, които не са алгебрични. Един *граф* $\mathbf{G}$ може да се мисли като структура, чийто универсум е множеството от върховете на графа, символи за константи и функ-

*Понякога структурите в алгебрата са снабдени с някаква наредба. Строго погледнато, такива структури не са алгебрични.

ционални символи няма и има единствен предикатен символ е $\mathtt{p}$, който е двуместен и за всеки два върха v_1 и v_2:

- ако си мислим, че $\mathtt{p}$ се интерпретира с предикат, то $\mathtt{p}^{\mathbf{G}}(v_1, v_2)$ е истина тогава и само тогава, когато има ребро от v_1 до v_2;
- ако си мислим, че $\mathtt{p}$ се интерпретира с релация, то $\langle v_1, v_2 \rangle \in \mathtt{p}^{\mathbf{G}}$ е истина тогава и само тогава, когато има ребро от v_1 до v_2, т. е.

$$\mathtt{p}^{\mathbf{G}} = \{\, \langle v_1, v_2 \rangle \mid \text{има ребро от } v_1 \text{ до } v_2 \,\}$$

3.1.9. Пример. Нека сигнатурата $\mathfrak{sig}$ е такава, че в нея има единствен символ за константа $\mathtt{c}$, единствен функционален символ $\mathtt{f}$, който е двуместен, и единствен предикатен символ $\mathtt{p}$, който също е двуместен. Нека структурата $\mathbf{M}$ за $\mathfrak{sig}$ е с универсум множеството на реалните числа, $\mathtt{c}^{\mathbf{M}} = 3$, $\mathtt{f}^{\mathbf{M}}(x, y) = x + y$ и $\mathtt{p}^{\mathbf{M}}(x, y)$ е съждението „$x < y$“. В такъв случай формулата

$$\mathtt{p}(\mathtt{c}, \mathtt{x}) \Rightarrow \mathtt{p}(\mathtt{c}, \mathtt{f}(\mathtt{x}, \mathtt{x}))$$

казва, че ако едно реално число x е по-голямо от 3, то $x + x$ също е по-голямо от 3.

✓ ***Задача* 33:** Какво казват в структурата от пример 3.1.9 следните формули:

$$\mathtt{p}(\mathtt{x}, \mathtt{y}) \,\&\, \mathtt{p}(\mathtt{y}, \mathtt{z}) \Rightarrow \mathtt{p}(\mathtt{x}, \mathtt{z}) \tag{6}$$

$$\mathtt{p}(\mathtt{x}, \mathtt{y}) \Rightarrow \mathtt{p}(\mathtt{f}(\mathtt{x}, \mathtt{z}), \mathtt{f}(\mathtt{y}, \mathtt{z})) \tag{7}$$

$$\mathtt{p}(\mathtt{c}, \mathtt{c}) \Rightarrow \mathtt{p}(\mathtt{x}, \mathtt{x}) \tag{8}$$

***Задача* 34:** В пример 3.1.8 видяхме, че графите може да се мислят като специален вид структури. Напишете формула, която е вярна в един граф тогава и само тогава, когато

а) графът е неориентиран;

б) графът е пълен

3.2. Оценки

В предния раздел дадохме дефиницията на структура, но все още не сме дали точна дефиниция за това какво значи една формула да бъде вярна в структура. В пример 3.1.9 разчитахме повече на интуицията си и здравия смисъл, отколкото на някаква точна математическа дефиниция.

За да дефинираме какво значи една формула да бъде вярна структура, е нужно да се научим да намираме стойността на термовете в структурата. За да пресметнем стойността на един терм обаче, не е достатъчно да знаем коя е структурата. Да разгледаме например терма $\mathtt{x} + \mathtt{x}$. Структурата ще ни каже какъв е смисълът на функционалният символ $+$, но не и стойността на променливата $\mathtt{x}$. А при различни стойности на променливата $\mathtt{x}$, стойността на този терм най-вероятно ще бъде различна. Затова ще дефинираме понятието „оценка на променливите“:

✓ **3.2.1. Дефиниция.** Функцията v е *оценка* в структурата $\mathbf{M}$, ако v съпоставя на всяка променлива елемент на универсума на $\mathbf{M}$.

Нека например структурата $\mathbf{M}$ е с универсум $\mathbb{R}$ и $+^{\mathbf{M}}$ е функцията събиране на реални числа. Нека оценката v в $\mathbf{M}$ е такава, че $v(\mathtt{x}) = 5$ и $v(\mathtt{y}) = 3$. Тогава стойността на терма $\mathtt{x} + \mathtt{y}$ в структурата $\mathbf{M}$ при оценка v трябва да бъде числото 8. Да забележим, че за да кажем каква трябва да бъде стойността на терма $\mathtt{x} + \mathtt{y}$, не бе нужно да знаем на колко е равно $v(\mathtt{z})$ за променливи $\mathtt{z}$, различни от $\mathtt{x}$ и $\mathtt{y}$. Единствените променливи, които имат отношение към стойността на терма $\mathtt{x} + \mathtt{y}$ са променливите, срещащи се в този терм, т.е. $\mathtt{x}$ и $\mathtt{y}$.

***Задача* 35:** Докажете, че за произволна структура $\mathbf{M}$ съществува поне една оценка в $\mathbf{M}$.

Точната дефиниция за стойност на терм е следната:

✓ **3.2.2. Дефиниция.** Нека $\mathbf{M}$ е структура за сигнатурата $\mathfrak{sig}$ и v е оценка в $\mathbf{M}$. *Стойността* в структурата $\mathbf{M}$ при оценка v на термовете при сигнатура $\mathfrak{sig}$ се дефинира индуктивно:

а) ако $\mathtt{x}$ е променлива, то стойността на терма $\mathtt{x}$ е $v(\mathtt{x})$;

б) ако $\mathtt{c}$ е символ за константа, то стойността на терма $\mathtt{c}$ е $\mathtt{c}^{\mathbf{M}}$;

в) ако $\mathtt{f}$ е n-местен функционален символ и $\tau_1, \tau_2, \ldots, \tau_n$ са термове, то стойността на терма $\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$ е равна на $\mathtt{f}^{\mathbf{M}}(\mu_1, \mu_2, \ldots, \mu_n)$, където μ_i е стойността на τ_i в $\mathbf{M}$ при оценка v.

Да обърнем внимание, че стойността на терм в структура е дефинирана само ако термът и структурата са за една и съща сигнатура. Например няма как да пресметнем стойността на терма $\mathtt{x}+\mathtt{y}$, ако структурата е за сигнатура, в която няма функционален символ $+$. Стойността на атомарна формула също е дефинирана само ако атомарната формула и структурата са за една и съща сигнатура.

3.2.3. Означение. Нека $\mathbf{M}$ е структура и τ е терм. *Смисълът* или *стойността* на терма τ в структурата $\mathbf{M}$ е функция, означавана с $[\![\tau]\!]^{\mathbf{M}}$, чийто аргумент е оценка v в $\mathbf{M}$, а стойността ѝ е равна на стойността на τ в структурата $\mathbf{M}$ при оценка v. Следователно можем да означаваме стойността на терм τ в структура $\mathbf{M}$ при оценка v посредством семантичните скоби $[\![\tau]\!]^{\mathbf{M}}v$.*

3.2.4. *Забележка*: Използвайки току-що даденото означение, можем да формулираме точките от дефиницията за стойност на терм по следния начин:

- стойността $[\![\mathtt{x}]\!]^{\mathbf{M}}v$ на променлива $\mathtt{x}$ се определя от оценката v и е равна на $v(\mathtt{x})$;
- стойността $[\![\mathtt{c}]\!]^{\mathbf{M}}$ на символ за константа $\mathtt{c}$ се определя от структурата;
- стойността на терм от вида $\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$ се получава като „спускаме“ рекурсивно семантичните скоби $[\![\cdot]\!]^{\mathbf{M}}$:

$$[\![\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)]\!]^{\mathbf{M}}v = [\![\mathtt{f}]\!]^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}v, [\![\tau_2]\!]^{\mathbf{M}}v, \ldots, [\![\tau_n]\!]^{\mathbf{M}}v)$$

Например

$$[\![\mathtt{g}(\mathtt{f}(\mathtt{x}, \mathtt{c}), \mathtt{y})]\!]^{\mathbf{M}}v = [\![\mathtt{g}]\!]^{\mathbf{M}}([\![\mathtt{f}]\!]^{\mathbf{M}}(v(\mathtt{x}), [\![\mathtt{c}]\!]^{\mathbf{M}}), v(\mathtt{y})) = \mathtt{g}^{\mathbf{M}}(\mathtt{f}^{\mathbf{M}}(v(\mathtt{x}), \mathtt{c}^{\mathbf{M}}), v(\mathtt{y}))$$

Едно важно свойство на стойността на терм е това, че стойността не зависи от променливите, които не се срещат в терма. Например за да пресметнем стойността на терма $\mathtt{x} + (\mathtt{y} + \mathtt{y})$, не е нужно да знаем каква стойност има променливата $\mathtt{z}$. Да докажем този факт.

✓ **3.2.5. Твърдение.** *Нека v и v' са оценки в структурата $\mathbf{M}$. Ако $v(\mathtt{x}) = v'(\mathtt{x})$ за всяка променлива $\mathtt{x}$, която се среща в терма τ, то $[\![\tau]\!]^{\mathbf{M}}v = [\![\tau]\!]^{\mathbf{M}}v'$.*

✓ Доказателство. С индукция по терма τ, използвайки изискванията на дефиницията за стойност на терм (вж. забележка 3.2.4).

Ако $\tau = \mathtt{x}$ е променлива, то $[\![\tau]\!]^{\mathbf{M}}v = [\![\mathtt{x}]\!]^{\mathbf{M}}v = v(\mathtt{x})$. Тъй като по условие v и v' съвпадат за променливи, срещащи се в τ (а в случая за $\mathtt{x} = \tau$), то последното е равно на $v'(\mathtt{x}) = [\![\mathtt{x}]\!]^{\mathbf{M}}v' = [\![\tau]\!]^{\mathbf{M}}v'$.

*Едно от често използваните в алгебрата означения за стойността на терм τ в структура $\mathbf{M}$ при оценка v е $\tau^{\mathbf{M}}[v]$. За съжаление в математическата логика се използват различни означения за стойността на терм. Означението $[\![\tau]\!]^{\mathbf{M}}v$, което използваме в този курс, е заимствано от теорията на езиците за програмиране.

Ако $\tau = \mathtt{c}$ е символ за константа, то както $[\![\tau]\!]^{\mathbf{M}}v = [\![\mathtt{c}]\!]^{\mathbf{M}}v$, така и $[\![\tau]\!]^{\mathbf{M}}v' = [\![\mathtt{c}]\!]^{\mathbf{M}}v'$ е равно на $\mathtt{c}^{\mathbf{M}}$.

Нека $\tau = \mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$. Тогава

$$[\![\tau]\!]^{\mathbf{M}}v = \mathtt{f}^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}v, [\![\tau_2]\!]^{\mathbf{M}}v, \ldots, [\![\tau_n]\!]^{\mathbf{M}}v)$$
$$[\![\tau]\!]^{\mathbf{M}}v' = \mathtt{f}^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}v', [\![\tau_2]\!]^{\mathbf{M}}v', \ldots, [\![\tau_n]\!]^{\mathbf{M}}v')$$

Изразите отдясно на равенствата обаче са равни, защото съгласно индукционното предположение $[\![\tau_i]\!]^{\mathbf{M}}v = [\![\tau_i]\!]^{\mathbf{M}}v'$. ∎

✓ ***Задача* 36:** Ако термът τ не съдържа нито една променлива, то за произволна структура $\mathbf{M}$ и оценки v и v' в $\mathbf{M}$, стойността на τ в структурата $\mathbf{M}$ при оценка v е равна на стойността на τ в $\mathbf{M}$ при оценка v'.

Тъй като стойността на терм зависи само от стойността на променливите, срещащи се в терма, то ще дефинираме понятието „частична оценка“.

3.2.6. Дефиниция. *Частична оценка* в структурата $\mathbf{M}$ е функция, чиято дефиниционна област съдържа само променливи, която на всяка променлива от дефиниционната си област съпоставя елемент от универсума на $\mathbf{M}$.

Разбира се, всяка оценка в $\mathbf{M}$ е и частична оценка в $\mathbf{M}$, но не всяка частична оценка в $\mathbf{M}$ е оценка в $\mathbf{M}$.

3.2.7. Дефиниция. Нека частичната оценка v в структура $\mathbf{M}$ е дефинирана за всички променливи, срещащи се в терм τ. Тогава стойността на τ в структурата $\mathbf{M}$ при частична оценка v ще дефинираме да бъде равна на стойността на τ при коя да е оценка v', която съвпада с v за променливите, срещащи се в τ. Да забележим, че съгласно твърдение 3.2.5 така дефинираната стойност не зависи от избора на v'. Стойността на τ в структура $\mathbf{M}$ при частична оценка v означаваме също както стойността при обикновена оценка: $[\![\tau]\!]^{\mathbf{M}}v$.

3.2.8. Означение. За произволни различни променливи $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ и елементи $\mu_1, \mu_2, \ldots, \mu_n$ на универсума на структура $\mathbf{M}$, с

$$[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \mu_1, \mu_2, \ldots, \mu_n]$$

ще означаваме частичната оценка v, чиято дефиниционна област е множеството $\{\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n\}$ и за която $v(\mathtt{x}_i) = \mu_i$ за всяко $i \in \{1, 2, \ldots, n\}$:

$$v(\xi) = \begin{cases} \mu_1, & \text{ако } \xi = \mathtt{x}_1 \\ \mu_2, & \text{ако } \xi = \mathtt{x}_2 \\ \ldots & \\ \mu_n, & \text{ако } \xi = \mathtt{x}_n \\ \text{недефинирано}, & \text{ако } \xi \notin \{\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n\} \end{cases}$$

3.2.9. Означение. а) Когато τ е терм, ще използваме записа

$$\tau(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$$

когато искаме да кажем, че променливите $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ са различни по между си и всички променливи, които се срещат в τ, са измежду $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$.

б) Нека $\tau(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$ е терм и $\mu_1, \mu_2, \ldots, \mu_n$ са произволни елементи на универсума на структурата $\mathbf{M}$. Ще пишем

$$[\![\tau]\!]^{\mathbf{M}}[\mu_1, \mu_2, \ldots, \mu_n]$$

вместо

$$[\![\tau]\!]^{\mathbf{M}}[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \mu_1, \mu_2, \ldots, \mu_n]$$

✓ ***Задача* 37:** Нека структурата $\mathbf{M}$ е с универсум множеството на реалните числа, двуместният функционален символ $\mathtt{f}$ се интерпретира като функцията събиране (т.е. $\mathtt{f}^{\mathbf{M}}(a, b) = a + b$ за произволни $a, b \in \mathbb{R}$) и двуместният функционален символ $\mathtt{g}$ се интерпретира като умножение (т.е. $\mathtt{g}^{\mathbf{M}}(a, b) = ab$ за произволни $a, b \in \mathbb{R}$). Намерете такъв терм $\tau(\mathtt{x}_1, \mathtt{x}_2)$, че

$$[\![\tau]\!]^{\mathbf{M}}[a, b] = a^2 + b$$

за произволни реални числа a и b.

3.3. Вярност на формула в структура

Да си припомним, че стойността на един терм в структура зависи от оценката, при която оценяваме терма. Това е така, защото структурата не дава стойност на променливите, които се срещат в терма. Също така да си припомним, че за да оценим един терм е достатъчно да знаем каква е стойността на променливите, които се срещат в него (твърдение 3.2.5).

Подобно се оказва и положението при формулите с тази разлика, че тук съществена е само стойността на свободните променливи. Да разгледаме например формулата

$$\forall \mathtt{x}\, \mathtt{p}(\mathtt{x}, \mathtt{y}, \mathtt{z}, \mathtt{t}) \tag{9}$$

За да може да се дефинира верността на тази формула е необходимо:

– структурата да каже кое е множеството, в което „работи“ кванторът $\forall \mathtt{x}$, т.е. универсумът;

– структурата да каже как се интерпретира предикатният символ $\mathtt{p}$;

– оценката да каже каква е стойността на променливите $\mathtt{y}$, $\mathtt{z}$ и $\mathtt{t}$. Стойността на останалите променливи, включително на $\mathtt{x}$, е без значение.

Нека например универсумът на структурата $\mathbf{M}$ е множеството на естествените числа и $\mathtt{p}^{\mathbf{M}}$ е предикатът

$$\mathtt{p}^{\mathbf{M}}(n, m, k, l) \longleftrightarrow m^{n+3} + m^{n+3} \neq l^{n+3}$$

В такъв случай формула (9) е вярна при оценка v тогава и само тогава, когато е вярно съждението:

За всяко естествено число n е вярно $a^{n+3} + b^{n+3} \neq c^{n+3}$

където $a = v(\mathtt{y})$, $b = v(\mathtt{z})$ и $c = v(\mathtt{t})$.

В общия случай, ако универсумът на структурата $\mathbf{M}$ е $|\mathbf{M}|$, тогава горната формула е вярна в $\mathbf{M}$ при оценка v тогава и само тогава, когато е вярно съждението:

За всеки елемент μ на $|\mathbf{M}|$ е вярно $\mathtt{p}^{\mathbf{M}}(\mu, v(\mathtt{y}), v(\mathtt{z}), v(\mathtt{t}))$.

Сега нека разгледаме по-общата ситуация на формула, в която след квантора стои не атомарна формула, а произволна формула

$$\forall \mathtt{x}\, \varphi$$

Кога трябва да считаме, че тази формула е вярна в структурата $\mathbf{M}$ при оценка v? Като пръв и най-естествен опит може да пробваме със следното съждение:

При всяка възможна стойност на $\mathtt{x}$ от универсума на $\mathbf{M}$ формулата φ е вярна.

Един проблем в тази формулировка е това, че в нея нищо не се казва за останлите променливи във φ и за оценката, която дава стойност на тези променливи. На втори опит получаваме следното:

При всяка възможна стойност на $\mathtt{x}$ *от универсума на* $\mathbf{M}$ *формулата* φ *е вярна при оценка* v*.*

В тази формулировка обаче възниква друг проблем. Какво означава „вярна при оценка v при еди-каква си стойност на променливата $\mathtt{x}$“? Та нали оценката v дава стойност на всички променливи, включително и на $\mathtt{x}$? На трети опит получаваме следното съждение:

За всеки елемент μ *на универсума на* $\mathbf{M}$ *формулата* φ *е вярна при модифицираната оценка* v'*, която се дефинира по следния начин:*

$$v'(\xi) = \begin{cases} \mu, & \text{ако } \xi = \mathtt{x} \\ v(\xi), & \text{иначе} \end{cases}$$

Именно на този вариант ще се спрем, когато дефинираме стойността на формула в структура при оценка. Преди това обаче нека дефинираме по-формално понятието „модифицирана оценка“.

✓ **3.3.1. Дефиниция.** Нека v е оценка в структурата $\mathbf{M}$, $\mathtt{x}$ е променлива и μ е елемент на универсума на $\mathbf{M}$. Тогава оценката

$$v'(\xi) = \begin{cases} \mu, & \text{ако } \xi = \mathtt{x} \\ v(\xi), & \text{иначе} \end{cases}$$

се нарича *модифицирана оценка* и ще бъде означавана така:

$$[\mathtt{x} \coloneqq \mu | v]$$

С други думи, модифицираната оценка $[\mathtt{x} \coloneqq \mu|v]$ е функция, която при аргумент $\mathtt{x}$ връща стойност μ, а за останалите аргументи съвпада с оценката v.

Използвайки модифицирани оценки, можем да дадем дефиницията за верността на формула по следния начин:

✓ **3.3.2. Дефиниция.** Нека $\mathbf{M}$ е структура. За всяка формула φ от сигнатурата на $\mathbf{M}$ и оценка v в $\mathbf{M}$ ще дефинираме съждение, което ще записваме

$$\mathbf{M} \vDash \varphi[v]$$

и ще четем така: „Формулата φ е вярна в структурата $\mathbf{M}$ при оценка v.“

Съждението $\mathbf{M} \vDash \varphi[v]$ се дефинира рекурсивно по формулата φ както следва:

а) ако $\varphi = \mathtt{p}(\tau_1, \tau_2, \ldots, \tau_n)$ е атомарна формула, то $\mathbf{M} \vDash \varphi[v]$ е съждението*

$$\mathtt{p}^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}v, [\![\tau_2]\!]^{\mathbf{M}}v, \ldots, [\![\tau_n]\!]^{\mathbf{M}}v)$$

б) $\mathbf{M} \vDash (\psi_1 \,\&\, \psi_2)[v]$ е съждението

$\mathbf{M} \vDash \psi_1[v]$ *и* $\mathbf{M} \vDash \psi_2[v]$

в) $\mathbf{M} \vDash (\psi_1 \vee \psi_2)[v]$ е съждението

$\mathbf{M} \vDash \psi_1[v]$ *или* $\mathbf{M} \vDash \psi_2[v]$

г) $\mathbf{M} \vDash (\psi_1 \Rightarrow \psi_2)[v]$ е съждението

ако $\mathbf{M} \vDash \psi_1[v]$, *то* $\mathbf{M} \vDash \psi_2[v]$

д) $\mathbf{M} \vDash \neg\psi[v]$ е съждението

не е вярно, че $\mathbf{M} \vDash \psi[v]$

е) $\mathbf{M} \vDash \forall\mathtt{x}\,\psi[v]$ е съждението

за всяко $\mu \in |\mathbf{M}|$ *е вярно* $\mathbf{M} \vDash \psi[\mathtt{x} := \mu|v]$

ж) $\mathbf{M} \vDash \exists\mathtt{x}\,\psi[v]$ е съждението

съществува такова $\mu \in |\mathbf{M}|$, *че* $\mathbf{M} \vDash \psi[\mathtt{x} := \mu|v]$

3.3.3. Дефиниция. Когато съждението $\mathbf{M} \vDash \varphi[v]$ не е вярно, казваме, че формулата φ не е вярна в структурата $\mathbf{M}$ при оценка v.

✓ **3.3.4. Пример.** Нека сигнатурата $\mathfrak{sig}$ е такава, че в нея има единствен символ за константа $\mathtt{c}$, единствен функционален символ $\mathtt{f}$, който е двуместен, и единствен предикатен символ $\mathtt{p}$, който също е двуместен. Нека структурата $\mathbf{M}$ за $\mathfrak{sig}$ е с универсум множеството на реалните числа и

$$\mathtt{c}^{\mathbf{M}} = 3$$
$$\mathtt{f}^{\mathbf{M}}(x, y) = x + y$$
$$\mathtt{p}^{\mathbf{M}}(x, y) \longleftrightarrow x < y$$

Нека освен това оценката v е такава, че $v(\mathtt{x}) = 35$ и $v(\mathtt{y}) = 13$. Тогава атомарната формула $\mathtt{p}(\mathtt{f}(\mathtt{x}, \mathtt{c}), \mathtt{y})$ не е вярна в структурата $\mathbf{M}$ при

*Ако искаме, може да си припомним, че $[\![\mathtt{p}]\!]^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}v, [\![\tau_2]\!]^{\mathbf{M}}v, \ldots, [\![\tau_n]\!]^{\mathbf{M}}v)$ е съкратен запис на

$$\langle [\![\tau_1]\!]^{\mathbf{M}}v, [\![\tau_2]\!]^{\mathbf{M}}v, \ldots, [\![\tau_n]\!]^{\mathbf{M}}v\rangle \in \mathtt{p}^{\mathbf{M}}$$

оценка v, защото

$$\begin{aligned}\mathbf{M} \vDash \mathtt{p}(\mathtt{f}(\mathtt{x},\mathtt{c}),\mathtt{y})[v] &\longleftrightarrow \mathtt{p}^{\mathbf{M}}(\mathtt{f}^{\mathbf{M}}(v(\mathtt{x}),\mathtt{c}^{\mathbf{M}}),v(\mathtt{y}))\\ &\longleftrightarrow \mathtt{p}^{\mathbf{M}}(\mathtt{f}^{\mathbf{M}}(35,3),13)\\ &\longleftrightarrow \mathtt{p}^{\mathbf{M}}(35+3,13)\\ &\longleftrightarrow 35+3<13 \longleftrightarrow \text{лъжа}\end{aligned}$$

✓ **3.3.5. Дефиниция.** Две формули φ и ψ са *еквивалентни*, ако при произволни структура $\mathbf{M}$ и оценка v в $\mathbf{M}$, съжденията $\mathbf{M} \vDash \varphi[v]$ и $\mathbf{M} \vDash \psi[v]$ са еквивалентни. Записваме това така:

$$\vDash \varphi \Leftrightarrow \psi$$

✓ **3.3.6. Следствие.**

а) $\vDash \varphi \Leftrightarrow \varphi$

б) ако $\vDash \varphi \Leftrightarrow \psi$, *то* $\vDash \psi \Leftrightarrow \varphi$

в) ако $\vDash \varphi \Leftrightarrow \psi$ *и* $\vDash \psi \Leftrightarrow \chi$, *то* $\vDash \varphi \Leftrightarrow \chi$

Доказателство. Следва непосредствено от дефиниция 3.3.5. ■

Еквивалентните формули не са задължително равни, нито дори конгруентни. Може да си мислим за тях като изрази, които винаги имат една и съща стойност. Също както изразите

$$(x+y)^2 \text{ и } x^2+2xy+y^2$$

са различни, но винаги са равни, така и еквивалентните формули дори да са различни като запис, винаги имат еднаква вярност. Да бъдат две формули еквивалентни означава, че смисълът им е един и същ във всяка структура и при всяка оценка. Ако две формули са еквивалентни, то без значение каква е структурата и каква е оценката, винаги ако едната от тези формули се окаже вярна, то и другата ще бъде вярна, и ако едната от тях е невярна, то и другата ще бъде невярна.

3.3.7. Пример. Формулите

$$\mathtt{p}(\mathtt{x}) \vee \mathtt{q}(\mathtt{f}(\mathtt{y})) \text{ и } \mathtt{q}(\mathtt{f}(\mathtt{y})) \vee \mathtt{p}(\mathtt{x})$$

са еквивалентни — без значение как структурата интерпретира символите `p` и `f` и как оценката оценява променливите `x` и `y`, винаги ако едната от тези формули се окаже вярна, то и другата ще бъде вярна, и ако едната от тях е невярна, то и другата ще бъде невярна.

Когато в някой аритметичен израз заменим някой подизраз с друг подизраз, който е равен, тогава и целият израз ще бъде равен. Например изразът

$$\oint_C (dx + (a+b)^2\, dy)$$

е равен на израза

$$\oint_C (dx + (a^2 + 2ab + b^2)\, dy)$$

като за да стигнем до този извод дори не е нужно да знаем какво значи криволинеен интеграл.* Нещо подобно е вярно и за еквивалентните формули и трябва да си го докажем.

✓ **3.3.8. Твърдение.** *Нека формулата φ има подформула ψ и формулата ψ е еквивалентна на ψ'. Ако формулата φ' се получава от φ като заменим подформулата ψ с ψ', то формулите φ и φ' са еквивалентни.*

Доказателство. С индукция по построението на формулата φ.

Ако $\psi = \varphi$, то като заменим ψ с еквивалентна на нея формула ψ', ще получим $\varphi' = \psi'$ и значи φ и φ' ще са еквивалентни. Остава да разгледаме случая когато $\psi \neq \varphi$.

Ако φ е атомарна, то единствената ѝ подформула е $\psi = \varphi$, а вече разгледахме този случай.

Ако $\varphi = \chi_1 \,\&\, \chi_2$ и $\psi \neq \varphi$, то ψ ще е подформула на χ_1 или χ_2. Нека за определеност ψ е подформула на χ_1 и като заменим ψ на ψ' от χ_1 получаваме χ_1'. Съгласно индукционното предположение χ_1 и χ_1' са еквивалентни. Тъй като $\varphi' = \chi_1' \,\&\, \chi_2$, от тук получаваме, че φ и φ' също са еквивалентни.**

Случаите когато $\varphi = \chi_1 \vee \chi_2$ и $\varphi = \chi_1 \Rightarrow \chi_2$ се разглеждат аналогично.

Ако $\varphi = \neg\chi$ и $\psi \neq \varphi$, то ψ ще е подформула на χ. Като заменим ψ на ψ' от χ получаваме χ'. Съгласно индукционното предположение χ и

*Всъщност и двата интеграла са равни на нула за произволна затворена крива C.

**По-подробно това се обосновава така. Щом при произволна структура $\mathbf{M}$ и оценка v, $\mathbf{M} \vDash \chi_1[v]$ е еквивалентно на $\mathbf{M} \vDash \chi_1'[v]$, значи

$$\begin{aligned}
\mathbf{M} \vDash \varphi[v] &\longleftrightarrow \mathbf{M} \vDash (\chi_1 \,\&\, \chi_2)[v] \\
&\longleftrightarrow \mathbf{M} \vDash \chi_1[v] \text{ и } \mathbf{M} \vDash \chi_2[v] \\
&\longleftrightarrow \mathbf{M} \vDash \chi_1'[v] \text{ и } \mathbf{M} \vDash \chi_2[v] \\
&\longleftrightarrow \mathbf{M} \vDash (\chi_1' \,\&\, \chi_2)[v] \\
&\longleftrightarrow \mathbf{M} \vDash \varphi'[v]
\end{aligned}$$

χ' са еквивалентни. Тъй като $\varphi' = \neg\chi'$, от тук получаваме, че φ и φ' също са еквивалентни.

Ако $\varphi = \forall \mathtt{x}\, \chi$ и $\psi \neq \varphi$, то ψ е подформула на χ. Като заменим ψ с ψ' от χ получаваме χ'. Съгласно индукционното предположение χ и χ' са еквивалентни. Тъй като $\varphi' = \forall \mathtt{x}\, \chi'$, то от тук получаваме, че φ и φ' също са еквивалентни.*

Случаят когато $\varphi = \exists \mathtt{x}\, \chi$ се разглежда аналогично. ∎

Също както за термовете определихме как да ги оценяваме при частична оценка (вж. дефиниция 3.2.7), полезно е същото да можем да правим и при формулите. По този начин ще може да използваме напр. записа

$$\mathbf{M} \vDash \varphi[5, 4]$$

когато искаме да кажем, че формулата φ е вярна в $\mathbf{M}$ когато стойността на някои, предварително уточнени променливи, е 5 и 6. Дефиницията може да бъде аналогична:

3.3.9. Дефиниция. Нека частичната оценка v в структура $\mathbf{M}$ е дефинирана за всички свободни променливи във формулата φ. Ще казваме, че φ е вярна в структурата $\mathbf{M}$ при частична оценка v, ако φ е вярна в $\mathbf{M}$ при коя да е оценка v', която съвпада с v за свободните променливи на φ. Ще означаваме това така: $\mathbf{M} \vDash \varphi[v]$.

За да бъде смислена току-що дадената дефиниция, е нужно да покажем, че верността на φ не зависи от конкретния избор на оценката v'. Това следва от следното твърдение:

✓ **3.3.10. Твърдение.** *Нека v и v' са оценки в структура $\mathbf{M}$. Ако v и v' съвпадат за всички свободни променливи на формулата φ, то $\mathbf{M} \vDash \varphi[v]$ е еквивалентно на $\mathbf{M} \vDash \varphi[v']$.*

*По-подробно това се обосновава така. Щом при произволна структура $\mathbf{M}$ и оценка v, $\mathbf{M} \vDash \chi[v]$ е еквивалентно на $\mathbf{M} \vDash \chi'[v]$, значи

$$\begin{aligned}
\mathbf{M} \vDash \varphi[v] &\longleftrightarrow \mathbf{M} \vDash \forall \mathtt{x}\, \chi[v] \\
&\longleftrightarrow \text{за всяко } \mu \in |\mathbf{M}| \text{ е вярно, че } \mathbf{M} \vDash \chi[\mathtt{x} \coloneqq \mu|v] \\
&\longleftrightarrow \text{за всяко } \mu \in |\mathbf{M}| \text{ е вярно, че } \mathbf{M} \vDash \chi'[\mathtt{x} \coloneqq \mu|v] \\
&\longleftrightarrow \mathbf{M} \vDash \forall \mathtt{x}\, \chi'[v] \\
&\longleftrightarrow \mathbf{M} \vDash \varphi'[v]
\end{aligned}$$

✓ Доказателство. С индукция по формулата φ. Ако φ е атомарна и има вида $\mathtt{p}(\tau_1, \tau_2, \ldots, \tau_n)$, то

$$\mathbf{M} \vDash \varphi[v] \longleftrightarrow \mathtt{p}^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}v, [\![\tau_2]\!]^{\mathbf{M}}v, \ldots, [\![\tau_n]\!]^{\mathbf{M}}v)$$

Съгласно твърдение 3.2.5, последното е еквивалентно на

$$\mathtt{p}^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}v', [\![\tau_2]\!]^{\mathbf{M}}v', \ldots, [\![\tau_n]\!]^{\mathbf{M}}v') \longleftrightarrow \mathbf{M} \vDash \varphi[v']$$

Ако $\varphi = \psi_1 \,\&\, \psi_2$, то

$$\mathbf{M} \vDash \varphi[v] \longleftrightarrow \mathbf{M} \vDash \psi_1[v] \text{ и } \mathbf{M} \vDash \psi_2[v]$$

Съгласно индукционното предположение, последното е еквивалентно на

$$\mathbf{M} \vDash \psi_1[v'] \text{ и } \mathbf{M} \vDash \psi_2[v'] \longleftrightarrow \mathbf{M} \vDash \varphi[v']$$

Когато φ има вида $\psi_1 \vee \psi_2$ или $\psi_1 \Rightarrow \psi_2$, може да разсъждаваме аналогично.

Ако $\varphi = \forall \mathtt{x}\, \psi$, то

$$\mathbf{M} \vDash \varphi[v] \longleftrightarrow \text{за всяко } \mu \in |\mathbf{M}| \text{ е вярно } \mathbf{M} \vDash \psi[\mathtt{x} := \mu|v]$$

и

$$\mathbf{M} \vDash \varphi[v'] \longleftrightarrow \text{за всяко } \mu \in |\mathbf{M}| \text{ е вярно } \mathbf{M} \vDash \psi[\mathtt{x} := \mu|v']$$

Съгласно индукционното предположение, изразите отдясно на тези две еквивалентности са еквивалентни. (Защо може да прилагаме индукционното предположение?)

Когато $\varphi = \exists \mathtt{x}\, \psi$, се разсъждава аналогично. ∎

✓ **3.3.11. Означение.**

а) Когато φ е формула, ще използваме записа

$$\varphi(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$$

когато искаме да кажем, че променливите $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ са различни по между си и всички свободни променливи на φ, са измежду $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$.

б) Нека $\varphi(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$ е формула и $\mu_1, \mu_2, \ldots, \mu_n$ са произволни елементи на универсума на структурата $\mathbf{M}$. Ще пишем

$$\mathbf{M} \vDash \varphi[\mu_1, \mu_2, \ldots, \mu_n]$$

вместо

$$\mathbf{M} \vDash \varphi[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n := \mu_1, \mu_2, \ldots, \mu_n]$$

✓ ***Задача* 38:** Нека структурата $\mathbf{M}$ е с универсум множеството на реалните числа, двуместният функционален символ $\mathtt{f}$ се интерпретира като функцията събиране (т.е. $\mathtt{f}^{\mathbf{M}}(a,b) = a+b$ за произволни $a, b \in \mathbb{R}$), двуместният функционален символ $\mathtt{g}$ се интерпретира като умножение (т. е. $\mathtt{g}^{\mathbf{M}}(a,b) = ab$ за произволни $a, b \in \mathbb{R}$) и двуместният предикатен символ $\mathtt{p}$ се интерпретира като равенство (т.е. $\mathtt{p}^{\mathbf{M}}(a,b) \longleftrightarrow a = b$ за произволни $a, b \in \mathbb{R}$). Намерете такива терм $\tau(\mathtt{x}_1, \mathtt{x}_2)$ и атомарна формула $\varphi(\mathtt{x}_1, \mathtt{x}_2)$, че

$$[\![\tau]\!]^{\mathbf{M}}[a,b] = a^2 + b$$
$$\mathbf{M} \vDash \varphi[a,b] \longleftrightarrow a^2 = 2b$$

за произволни реални числа a и b.

Хубаво е, че сме дефинирали как се прилага субституция към произволен терм, обаче ползата от това не би била голяма, ако стойността при оценка v на терма след прилагане на субституцията не е такава, каквато трябва. Да приложим субституция s означава да заменим всяка променлива $\mathtt{x}$ с $s(\mathtt{x})$. Нека оценката w дава на променливата $\mathtt{x}$ стойност равна на стойността при оценка v на $s(\mathtt{x})$. Тогава може да очакваме, че стойността на τ при оценка w е равна на стойността на τs при първоначалната оценка v. Следващото твърдение показва, че това е така. След това, в твърдение 3.3.17, ще видим че подобно твърдение е вярно и за формулите.

3.3.12. Твърдение. *Нека s е произволна субституция, а v произволна оценка в някоя структура $\mathbf{M}$. Да дефинираме оценката w по следния начин:*

$$w(\mathtt{z}) = [\![\mathtt{z}s]\!]^{\mathbf{M}} v$$

Тогава за всеки терм τ

$$[\![\tau s]\!]^{\mathbf{M}} v = [\![\tau]\!]^{\mathbf{M}} w$$

Доказателство. С индукция по терма τ. Ако термът $\tau = \mathtt{x}$ е променлива, то получаваме исканото от дефиницията на w:

$$[\![\tau s]\!]^{\mathbf{M}} v = [\![\mathtt{x}s]\!]^{\mathbf{M}} v = w(\mathtt{x}) = [\![\tau]\!]^{\mathbf{M}} w$$

Ако термът $\tau = \mathtt{c}$ е символ за константа, то получаваме исканото съвсем лесно:

$$[\![\tau s]\!]^{\mathbf{M}} v = [\![\mathtt{c}s]\!]^{\mathbf{M}} v = [\![\mathtt{c}]\!]^{\mathbf{M}} v = \mathtt{c}^{\mathbf{M}} = [\![\tau]\!]^{\mathbf{M}} w$$

Ако $\tau = \mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$, то получаваме исканото, използвайки индукционното предположение за $\tau_1, \tau_2, \ldots, \tau_n$:

$$\begin{aligned}
[\![\tau s]\!]^{\mathbf{M}} v &= [\![\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)s]\!]^{\mathbf{M}} v \\
&= [\![\mathtt{f}(\tau_1 s, \tau_2 s, \ldots, \tau_n s)]\!]^{\mathbf{M}} v && \text{(деф. 2.7.2)} \\
&= \mathtt{f}^{\mathbf{M}}([\![\tau_1 s]\!]^{\mathbf{M}} v, [\![\tau_2 s]\!]^{\mathbf{M}} v, \ldots, [\![\tau_n s]\!]^{\mathbf{M}} v) && \text{(деф. 3.2.2 в))} \\
&= \mathtt{f}^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}} w, [\![\tau_2]\!]^{\mathbf{M}} w, \ldots, [\![\tau_n]\!]^{\mathbf{M}} w) && \text{(инд. предп.)} \\
&= [\![\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)]\!]^{\mathbf{M}} w && \text{(деф. 3.2.2 в))} \\
&= [\![\tau]\!]^{\mathbf{M}} w
\end{aligned}$$

■

Да си припомним, че записът $\tau(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$ означава, че променливите $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ са различни по между си и всички променливи, срещащи се в τ, са измежду $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$. Освен това, ако $\sigma_1, \sigma_2, \ldots, \sigma_n$ също са термове (с променливи не задължително измежду $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$), тогава термът $\tau[\sigma_1, \sigma_2, \ldots, \sigma_n]$ се получава като заместим в τ всяко срещане на променлива измежду $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ със съответния терм от $\sigma_1, \sigma_2, \ldots, \sigma_n$. Използвайки тези означения, може да преформулираме току-що доказаното твърдение по следния начин:

✓ **3.3.13. Твърдение (лема за субституциите за термове).** *Нека $\tau(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n), \sigma_1, \sigma_2, \ldots, \sigma_n$ са термове и v е оценка в структурата* $\mathbf{M}$. *Тогава*

$$[\![\tau[\sigma_1, \sigma_2, \ldots, \sigma_n]]\!]^{\mathbf{M}} v = [\![\tau]\!]^{\mathbf{M}}[[\![\sigma_1]\!]^{\mathbf{M}} v, [\![\sigma_2]\!]^{\mathbf{M}} v, \ldots, [\![\sigma_n]\!]^{\mathbf{M}} v]$$

Преди да докажем аналогично твърдение за произволни формули, ще докажем като лема частния случай когато няма квантори с опасни променливи.

3.3.14. Лема. *Нека φ е формула. Ако субституцията s е такава, че φ не съдържа квантори с опасни променливи, то за произволна оценка v в структура* $\mathbf{M}$ *може да дефинираме нова оценка w*

$$w(\mathtt{z}) = [\![\mathtt{z}s]\!]^{\mathbf{M}} v$$

и, ако направим това, тогава

$$\mathbf{M} \vDash \varphi s[v] \longleftrightarrow \mathbf{M} \vDash \varphi[w]$$

Тъй като искаме да отъждествяваме конгруентните формули, трябва да докажем, че стойностите на кои да е две конгруентни формули са еквивалентни.

✓ **3.3.15. Твърдение.** *Ако две формули са конгруентни, то те са еквивалентни.*

Доказателство. Нека $\varphi \equiv \psi$. Тогава формулите φ и ψ ще имат равен брой символи. Ще докажем, че тези формули са еквивалентни с пълна математическа индукция по броя на символите в тях.

Както обикновено, да разгледаме случаи според вида на формулите. Ако φ е атомарна, то $\varphi = \psi$, защото конгруентните формули могат да се различават само при свързаните променливи, а атомарните формули не съдържат свързани променливи. Ако $\varphi = \bot$, то положението е същото.

Ако $\varphi = \varphi' \,\&\, \varphi''$, то от дефиницията за конгруентност ще следва, че ψ има вида $\psi' \,\&\, \psi''$, където $\varphi' \equiv \psi'$ и $\varphi'' \equiv \psi''$. Съгласно индукционното предположение $\vDash \varphi' \Leftrightarrow \psi'$ и $\vDash \varphi'' \Leftrightarrow \psi''$ и значи от твърдение 3.3.8 получаваме, че $\varphi' \,\&\, \varphi'' \equiv \psi' \,\&\, \psi''$.

Случаите когато $\varphi = \varphi' \vee \varphi''$ и $\varphi = \varphi' \Rightarrow \varphi''$ се разглеждат аналогично.

Ако $\varphi = \forall \mathtt{x}\, \varphi'$, то от дефиницията за конгруентност ще следва, че ψ има вида $\forall \mathtt{y}\, \psi'$. Не можем директно да сведем към индукционното предположение, защото променливите $\mathtt{x}$ и $\mathtt{y}$ са свободни съответно в φ' и ψ, така че е възможно φ' и ψ' да не са конгруентни. Затова ще постъпим по-заобиколно. Нека $\mathtt{z}$ е променлива, която не се среща никъде — нито в φ, нито в ψ. Ако φ'' се получава от φ' като заменим всяко свободно срещане променливата $\mathtt{x}$ с $\mathtt{z}$ и аналогично ψ'' от ψ' като заменим $\mathtt{y}$ с $\mathtt{z}$, то $\varphi = \forall \mathtt{x}\, \varphi' \equiv \forall \mathtt{z}\, \varphi''$ и $\psi = \forall \mathtt{y}\, \psi' \equiv \forall \mathtt{z}\, \psi''$. Но $\varphi \equiv \psi$, значи $\forall \mathtt{z}\, \varphi'' \equiv \forall \mathtt{z}\, \psi''$, от където лема 2.6.13 ще ни даде $\varphi'' \equiv \psi''$ и значи може да приложим индукционното предположение за φ'' и ψ'' и да получим $\vDash \varphi'' \Leftrightarrow \psi''$. Съгласно твърдение 3.3.8, $\vDash \forall \mathtt{z}\, \varphi'' \Leftrightarrow \forall \mathtt{z}\, \psi''$.

Остава да установим, че $\vDash \forall \mathtt{x}\, \varphi' \Leftrightarrow \forall \mathtt{z}\, \varphi''$ и $\vDash \forall \mathtt{y}\, \psi' \Leftrightarrow \forall \mathtt{z}\, \psi''$. Ще докажем първата от тези две еквивалентности; втората се доказва аналогично. Първо, да забележим, че в φ' няма квантори с опасни променливи при субституция $[\mathtt{x} \coloneqq \mathtt{z}]$, защото единствената променлива, която може да бъде опасна е $\mathtt{z}$, а пък избрахме $\mathtt{z}$ да бъде напълно нова променлива. Следователно тази субституция се прилага към φ' просто като заменим всяко свободно срещане на $\mathtt{x}$ с $\mathtt{z}$. С други думи, $\varphi'' = \varphi'[\mathtt{x} \coloneqq \mathtt{z}]$.

Да изберем произволни структура $\mathbf{M}$ и оценка v в $\mathbf{M}$. Тогава от една

страна

$$\begin{aligned}\mathbf{M} \vDash \forall \mathtt{z}\, \varphi''[v] &\longleftrightarrow \mathbf{M} \vDash \forall \mathtt{z}\, (\varphi'[\mathtt{x} \coloneqq \mathtt{z}])[v] \\ &\longleftrightarrow \text{за всяко } \mu \in |\mathbf{M}|\ \mathbf{M} \vDash \varphi'[\mathtt{x} \coloneqq \mathtt{z}][\mathtt{z} \coloneqq \mu|v] \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(деф. 3.3.2)} \\ &\longleftrightarrow \text{за всяко } \mu \in |\mathbf{M}|\ \mathbf{M} \vDash \varphi'[\mathtt{x}, \mathtt{z} \coloneqq \mu, \mu|v] \quad \text{(лема 3.3.14)}\end{aligned}$$

От друга страна

$$\begin{aligned}\mathbf{M} \vDash \varphi[v] &\longleftrightarrow \mathbf{M} \vDash \forall \mathtt{x}\, \varphi'[v] \\ &\longleftrightarrow \text{за всяко } \mu \in |\mathbf{M}|\ \mathbf{M} \vDash \varphi'[\mathtt{x} \coloneqq \mu|v] \qquad \text{(деф. 3.3.2)}\end{aligned}$$

Тъй като променливата $\mathtt{z}$ не се среща в φ', то съгласно твърдение 3.3.10 $\mathbf{M} \vDash \varphi'[\mathtt{x}, \mathtt{z} \coloneqq \mu, \mu|v]$ е еквивалентно на $\mathbf{M} \vDash \varphi'[\mathtt{x} \coloneqq \mu|v]$.

Случаят $\varphi = \exists \mathtt{x}\, \varphi'$ се разглежда аналогично. ∎

Вече сме готови да докажем общия случай на лема 3.3.14.

3.3.16. Твърдение. *Нека φ е формула, s е субституция и v е оценка в структура $\mathbf{M}$. Ако оценката w е такава, че за всяка променлива $\mathtt{z}$*

$$w(\mathtt{z}) = [\![\mathtt{z}s]\!]^{\mathbf{M}} v$$

то

$$\mathbf{M} \vDash \varphi s[v] \longleftrightarrow \mathbf{M} \vDash \varphi[w]$$

Доказателство. Нека ψ е конгруентна с φ формула, която не съдържа квантори с опасни при субституция s променливи (такава формула има съгласно лема 2.6.11). Съгласно лема 3.3.14, $\mathbf{M} \vDash \psi[w]$ е еквивалентно на $\mathbf{M} \vDash (\psi s)[v]$. От тук получаваме исканото, защото формулата φ е конгруентна, а значи и еквивалентна с ψ, а от твърдение 2.6.9 имаме, че формулата φs е конгруентна, а значи и еквивалентна с формулата ψs. ∎

Използвайки други означения, може да преформулираме това твърдение по следния начин:

✓ **3.3.17. Твърдение (лема за субституциите за формули).** *Нека $\varphi(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$ е формула, $\tau_1, \tau_2, \ldots, \tau_n$ са термове и v е оценка в структурата $\mathbf{M}$. Тогава*

$$\mathbf{M} \vDash \varphi[\tau_1, \tau_2, \ldots, \tau_n][v] \longleftrightarrow \mathbf{M} \vDash \varphi[[\![\tau_1]\!]^{\mathbf{M}} v, [\![\tau_1]\!]^{\mathbf{M}} v, \ldots, [\![\tau_n]\!]^{\mathbf{M}} v]$$

3.4. Тъждествена вярност, изпълнимост, следване

Обикновено когато пишем някоя формула ние не си мислим, че променливите в нея имат някакви конкретни стойности. Например формулата

$$\mathtt{x} + \mathtt{y} = \mathtt{y} + \mathtt{x}$$

ни казва, че двуместната функция, означена с „+“, е комутативна, т.е. $\mathtt{x} + \mathtt{y} = \mathtt{y} + \mathtt{x}$ при произволни стойности на $\mathtt{x}$ и $\mathtt{y}$. Това показва, че понятието „вярност в структура при оценка“, което вече дефинирахме, не ни дава точно това, което искаме. Например горната формула ще бъде вярна в някоя структура при определена оценка тогава и само тогава, когато $\mathtt{x} + \mathtt{y} = \mathtt{y} + \mathtt{x}$ е вярно при някакви конкретни стойности на променливите $\mathtt{x}$ и $\mathtt{y}$, определени от оценката, а не при произволни стойности.*

Когато ни е дадена само структура, но не и оценка, една формула може да има различни „степени на вярност“. Например:

- формулата може да бъде вярна при всяка оценка;
- формулата може да бъде вярна при поне една оценка;
- формулата може да бъде невярна при произволна оценка.

Когато за една формула кажем просто, че е вярна, без да уточним при коя оценка, удобно е да считаме, че формулата е вярна при всяка оценка. Затова можем да дадем следната дефиниция:

✓ **3.4.1. Дефиниция.** а) Формулата φ е *(тъждествено) вярна* в структура $\mathbf{M}$, ако е вярна в структурата $\mathbf{M}$ при произволна оценка v.** Записваме това така:

$$\mathbf{M} \vDash \psi$$

б) Формулата φ е *тъждествено вярна* или *(предикатна) тавтология*, ако φ е тъждествено вярна във всяка структура.*** Записваме това така:

$$\vDash \psi$$

*Нека структурата е с универсум положителните реални числа и функционалният символ „+“ е интерпретиран като операцията деление. Тогава горната формула не е вярна, защото например $2/3 \neq 3/2$. Ако обаче оценката е такава, че $v(\mathtt{x}) = v(\mathtt{y}) = 42$, тогава горната формула ще бъде вярна при тази оценка, защото $42/42 = 42/42$.

**Други термини, които означават същото като „тъждествено вярна“ са *общовалидна* и *валидна*. На английски universally valid и valid.

***За предикатните тавтологии се използват още термините *общовалидна формула* и *валидна формула*.

Когато някоя формула е вярна не при всяка оценка, а при някоя оценка, или не във всяка структура, а в някоя структура, използваме термина „изпълнима формула“.

✓ **3.4.3. Дефиниция.** а) Формулата φ е *изпълнима* в структурата $\mathbf{M}$, ако съществува оценка, при която φ е вярна в $\mathbf{M}$.

б) Формулата φ е *изпълнима*, ако съществуват структура и оценка, при които φ е вярна.

Забележка: За съжаление използваната терминология на български е малко объркваща. Вместо „изпълнима“ на английски се използва прилагателното satisfiable. На това прилагателно съответства глаголът satisfy, който на български превеждаме с напълно различна дума — „удовлетворява“. Вместо „удовлетворима формула“ казваме „изпълнима формула“ и вместо (напр. в контекста на логическото програмиране) да казваме, че дадена цел се изпълнява, казваме, че целта се удовлетворява. На английски обаче във всеки един от тези случи се използва една и съща дума.

✓ **3.4.4. Дефиниция.**

а) Формула φ е *тъждествено невярна* в структурата $\mathbf{M}$, ако при всяка оценка v в $\mathbf{M}$, формулата φ е невярна в $\mathbf{M}$ при оценка v.

б) Формулата φ е *неизпълнима* в структурата $\mathbf{M}$, ако не съществува оценка, при която φ е вярна в $\mathbf{M}$.

в) Формула φ е *тъждествено невярна*, ако при всяка структура $\mathbf{M}$ и оценка v в $\mathbf{M}$, формулата φ е невярна в $\mathbf{M}$ при оценка v.

г) Формулата φ е *неизпълнима*, ако не съществуват структура и оценка, при които φ е вярна.

Следващите две задачи показват, че макар понятията от току-що дадената дефиниция да са естествени, те по същество не ни дават нищо ново.

✓ ***Задача* 39:** Нека φ е формула и $\mathbf{M}$ е структура. Докажете, че следните три неща са еквивалентни:

- φ не е изпълнима в $\mathbf{M}$;
- φ е неизпълнима в $\mathbf{M}$;
- φ е тъждествено невярна в $\mathbf{M}$.

✓ ***Задача* 40:** Нека φ е формула. Докажете, че следните три неща са еквивалентни:

– φ не е изпълнима;

– φ е неизпълнима;

– φ е тъждествено невярна.

✓ ***Задача* 41:** Възможно ли е една формула да не е тъждествено вярна в някоя структура, а пък да бъде изпълнима в същата структура?

***Задача* 42:** Възможно ли е една формула да бъде едновременно неизпълнима и тъждествено вярна в някоя структура? Защо при отговора на този въпрос се налага да използваме това, че универсумът на структурите е непразно множество?

✓ **3.4.5. Дефиниция.** Ако формула φ е тъждествено вярна в структура $\mathbf{M}$ (т.е. $\mathbf{M} \vDash \varphi$), казваме, че $\mathbf{M}$ е модел за φ. Ако всеки елемент на множество Γ от формули е тъждествено верен в структура $\mathbf{M}$, казваме, че $\mathbf{M}$ е модел за Γ и пишем $\mathbf{M} \vDash \Gamma$.

3.4.6. *Забележка*: Понякога в текстове, които не са писани от логици, думата „модел“ се приема за синоним на „структура“. Това е неправилно. Когато кажем, че $\mathbf{M}$ е модел, това означава не само, че $\mathbf{M}$ е структура, но също и че определени (известни може би от контекста) неща са верни в $\mathbf{M}$.

***Задача* 43:** Докажете, че ако някое множество от формули има поне един модел, то то има безброй много модели.

✓ **3.4.7. Дефиниция.** *Затворена формула* означава формула, която не съдържа свободни променливи.

3.4.8. Съгласно твърдение 3.3.10 верността на една затворена формула в структура $\mathbf{M}$ не зависи по никакъв начин от оценката, защото всеки две оценки съвпадат за свободните променливи на една затворена формула. Следователно има смисъл когато говорим за затворена формула да четем „$\mathbf{M} \vDash \varphi$“ като „$\varphi$ е вярна в $\mathbf{M}$“, изпускайки наречието „тъждествено“.

✓ **3.4.9. Твърдение.** *Ако φ е затворена формула, а $\mathbf{M}$ — произволна структура, то за всяка оценка v в $\mathbf{M}$*

$$\mathbf{M} \vDash \varphi[v] \longleftrightarrow \mathbf{M} \vDash \varphi \longleftrightarrow \varphi \textit{ е изпълнима в } \mathbf{M}$$

✓ Доказателство. Да припомним, че $\mathbf{M} \vDash \varphi$ означава, че формулата φ е тъждествено вярна в $\mathbf{M}$, т.е. φ е вярна при всяка оценка, а φ да

бъде изпълнима в $\mathbf{M}$ означава, че формулата φ е вярна в $\mathbf{M}$ при някоя оценка.

Съгласно твърдение 3.3.10 за кои да е две оценки v_1 и v_2 в $\mathbf{M}$ съждението $\mathbf{M} \vDash \varphi[v_1]$ е еквивалентно на $\mathbf{M} \vDash \varphi[v_2]$. Следователно φ е изпълнима в $\mathbf{M}$ тогава и само тогава, когато $\mathbf{M} \vDash \varphi[v]$ е истина при някоя оценка v, тогава и само тогава, когато $\mathbf{M} \vDash \varphi[v]$ е истина за всяка оценка v, тогава и само тогава, когато $\mathbf{M} \vDash \varphi$. ∎

***Задача* 44:** Възможно ли е една затворена формула да бъде изпълнима, но да не бъде тъждествено вярна?

Полезно е едно понятие, което казва, че от някакви формули следва някоя формула. Нека например множеството Γ съдържа аксиомите за полупръстен. Може да считаме, че формулата φ следва от Γ, ако φ е вярна във всички полупръстени. Следната дефиниция формализира интуицията на тази забележка:

✓ **3.4.10. Дефиниция.** Нека Γ е множество от формули. Казваме, че формулата φ *следва* от Γ, ако φ е тъждествено вярна във всяка структура, в която са тъждествено верни формулите в Γ. В този случай казваме още и накратко „от Γ следва φ“ и използваме следното означение:

$$\Gamma \vDash \varphi$$

3.4.11. ***Забележка*:** Нека не се объркваме и обърнем внимание, че използваме знака $\vDash$ за няколко различни цели. Когато Γ е множество от формули, тогава $\Gamma \vDash \varphi$ означава „от формулите в Γ следва φ“. Когато $\mathbf{M}$ е структура, $\mathbf{M} \vDash \varphi$ означава „φ е (тъждествено) вярно в $\mathbf{M}$“. А когато $\mathbf{M}$ е структура и v е оценка, $\mathbf{M} \vDash \varphi[v]$ означава „φ е вярна в $\mathbf{M}$ при оценка v“.

✓ ***Задача* 45:** Нека Γ и Δ са множества от формули и φ е формула. Докажете, че:

а) $\varphi \in \Gamma \longrightarrow \Gamma \vDash \varphi$

б) $\Delta \subseteq \Gamma$ и $\Delta \vDash \varphi \longrightarrow \Gamma \vDash \varphi$

в) Ако всеки елемент на Δ следва от Γ и $\Delta \vDash \varphi$, то $\Gamma \vDash \varphi$.

***Забележка*:** Условия а) и в) от задача 45 аксиоматизират това, което в абстрактната алгебрична логика се нарича *релация на следване*.*

*На английски consequence relation.

Да забележим също, че не е възможно да прилагаме безброй много конверсии, чийто ранг е 1 или 2. Наистина, вече видяхме, че не можем да прилагаме безброй конверсии само от ранг 1, следователно ако прилагаме само конверсии от ранг 1 или 2, ще трябва да прилагаме безброй пъти конверсии от ранг 2. Това обаче е невъзможно, защото след всяка конверсия от ранг 2 вторият член на редицата намалява, а това не може да се случва до безкрайност (да забележим, че конверсиите от ранг 1 не променят втория член на редицата). Това означава, че както и да прилагаме конверсиите, след краен брой стъпки или ще стигнем до редица, съдържаща само нули, или ще стигнем до конверсия от ранг поне 3.

Обаче не е възможно също да прилагаме безброй много конверсии, чийто ранг е 1, 2 или 3. Наистина, вече видяхме, че не можем да прилагаме безброй конверсии само от рангове 1 или 2, следователно ако прилагаме само конверсии от ранг 1, 2 или 3, ще трябва да прилагаме безброй пъти конверсии от ранг 3. Това обаче е невъзможно, защото след всяка конверсия от ранг 3 третият член на редицата намалява, а това не може да се случва до безкрайност (да забележим, че конверсиите от рангове 1 или 2 не променят третия член на редицата). Това означава, че както и да прилагаме конверсиите, след краен брой стъпки или ще стигнем до редица, съдържаща само нули, или ще стигнем до конверсия от ранг поне 4.

Разсъждавайки по подобен начин, можем да стигнем до извода, че както и да прилагаме конверсиите, след краен брой стъпки или ще стигнем до редица, съдържаща само нули, или ще стигнем до конверсия от ранг поне k, където k е произволно предварително избрано естествено число. Нека числото k е такова, че всички ненулеви членове в $(a_n)_{n=1}^{\infty}$ са измежду $a_1, a_2, \ldots, a_{k-1}$. В такъв случай няма да бъде възможна конверсия от ранг k или по-голям, следователно както и да прилагаме конверсиите, със сигурност ще стигнем до редица, съдържаща само нули. ■

✓ **3.7.7. Дефиниция.** Казваме, че дадена формула е в *отрицателна нормална форма*, ако:

- във формулата няма други логически операции, освен конюнкция ($\&$), дизюнкция ($\vee$), отрицание ($\neg$) и кванторите за всеобщност ($\forall$) и съществуване ($\exists$);
- всички отрицания във формулата се намират пред атомарни подформули.

✓ **3.7.8. Твърдение.** *За всяка формула може да се намери еквивалентна на нея (според класическата логика) формула, която е в отрицателна нормална форма.*

✓ Доказателство. За краткост, навсякъде в това доказателство където се говори за импликации, ще имаме предвид импликации, които не са отрицания, т. е. от вида $\varphi \Rightarrow \psi$, където $\psi \neq \bot$.

Нека φ е произволна формула. Най-напред да елиминираме импликациите във φ по следния начин: да заменяме докато може всяка подформула във φ, имаща вида $\psi' \Rightarrow \psi''$, където $\psi'' \neq \bot$, с формулата $\neg\psi' \vee \psi''$. Съгласно твърдение 3.7.3 след всяка такава замяна получаваме еквивалентна формула. Тъй като след всяка замяна броят на импликациите, намалява, то след краен брой стъпки ще получим формула без импликации, което значи, че във формулата няма други логически операции, освен конюнкция ($\&$), дизюнкция ($\vee$), отрицание ($\neg$) и кванторите за всеобщност ($\forall$) и съществуване ($\exists$).

Остава да „преместим“ отрицанията пред атомарни формули. За целта да прилагаме докато може замени от следния вид:

$$\neg\neg\psi \longmapsto \psi \tag{12}$$
$$\neg(\psi' \vee \psi'') \longmapsto \neg\psi' \,\&\, \neg\psi'' \tag{13}$$
$$\neg(\psi' \,\&\, \psi'') \longmapsto \neg\psi' \vee \neg\psi'' \tag{14}$$
$$\neg\exists \mathtt{x}\, \psi \longmapsto \forall \mathtt{x}\, \neg\psi \tag{15}$$
$$\neg\forall \mathtt{x}\, \psi \longmapsto \exists \mathtt{x}\, \neg\psi \tag{16}$$

Съгласно твърдения 3.7.1 г), 3.7.1 а), 3.7.1 б), 3.7.2 а) и 3.7.2 б), след всяка такава замяна получаваме еквивалентна формула. Ако след краен брой стъпки получим формула, към която не можем да приложим никоя от тези замени, то това означава, че сме стигнали до формула в отрицателна нормална форма. Това ни дава т. н. частична коректност на алгоритъма за привеждане в отрицателна нормална форма. Остава да видим защо този алгоритъм никога не се зацикля, т.е. трябва да докажем, че можем да прилагаме замени от горния вид само краен брой пъти. Това ще направим по два начина.

(I начин) Ще използваме фундираността на конверсията (лема 3.7.6).

Нека χ е произволна формула. *Височина* на χ ще наричаме дължината на най-дългата редица от вида

$$\chi_1, \chi_2, \chi_3, \ldots, \chi_k$$

където $\chi_1 = \chi$, $\chi_i \neq \chi_{i+1}$ и χ_{i+1} е подформула на χ_i. Ако си мислим формулата като дърво, тогава височината ѝ е равна на дължината на най-дългия клон в това дърво.

За всяка формула φ ще дефинираме редица $(a_n)_{n=1}^{\infty}$, която ще наречем *редица на* φ, по следния начин: нека a_i е равно на броя на подформулите на φ, които са от вида $\neg\chi$ и са с височина точно i.

Може да се забележи, че ако формулата φ' се получава от φ посредством някоя от замените (12)–(16), тогава редицата на φ' може да се получи от редицата на φ посредством конверсия. Съгласно лемата за фундираност на конверсията, след краен брой стъпки или ще стигнем формула, към която не можем да приложим никоя от замените (12)–(16), или ще стигнем формула, чиято редица се състои само от нули. Последното би означавало, че във формулата няма нито едно отрицание, но в този случай също е невъзможно да приложим замените (12)–(16).

(II начин) Да разгледаме следния начин, по който от формула получаваме аритметичен израз. Да заменим всяка атомарна формула с числото две. Да заменим всяка подформула от вида $\psi' \,\&\, \psi''$, $\psi' \vee \psi''$ и $\psi' \Rightarrow \psi''$ с $x + y$, където x и y са изразите, получени съответно от ψ' и ψ''. Да заменим всяка подформула от вида $\neg\psi$ с x^2, където x е изразът, получен от ψ. Да заменим всяка подформула от вида $\forall \mathtt{x}\, \psi$ и $\exists \mathtt{x}\, \psi$ с $1 + x$, където x е изразът, получен от ψ. Може да се забележи, че така полученият израз е естествено число, не по-малко от 2. Тъй като за всеки естествени числа по-големи или равни на 2 е изпълнено $(x^2)^2 > x$, $(x + y)^2 > x^2 + y^2$ и $(1 + x)^2 > 1 + x^2$, то стойността на получения израз ще намалява след всяка замяна от горния вид, а това не може да продължи до безкрайност. ∎

3.7.9. ***Забележка:*** Когато използваме първия начин за да докажем, че алгоритъмът за привеждане в отрицателна нормална форма спира, не е нужно да се използва толкова сложно твърдение като фундираността на конверсията. Нека редицата, съответстваща на някоя формула е $(a_n)_{n=1}^{\infty}$ и да разгледаме числото

$$\sum_{i=1}^{\infty} a_i 3^i$$

(сумата е коректна, защото само краен брой от числата a_i са ненулеви). Докато при дракона Мушмаху когато някоя негова глава бъде отсечена, може да му пораснат произволен брой нови глави, то тук, когато

вкараме някое отрицание „навътре“ във формулата, то се заменя най-много с две „по-малки“ отрицания. Това означава, че след всяка замяна от вида (12)–(16), така дефинираното число ще намалява, а това не може да продължи до безкрайност.

3.7.10. Твърдение. *За произволни формули φ и ψ, ако* $\mathtt{x}$ *не е свободна променлива на φ, то:*

а) $\models \varphi \,\&\, \exists \mathtt{x}\, \psi \Leftrightarrow \exists \mathtt{x}\, (\varphi \,\&\, \psi)$ $u \models \exists \mathtt{x}\, \psi \,\&\, \varphi \Leftrightarrow \exists \mathtt{x}\, (\psi \,\&\, \varphi)$;

б) $\models \varphi \,\&\, \forall \mathtt{x}\, \psi \Leftrightarrow \forall \mathtt{x}\, (\varphi \,\&\, \psi)$ $u \models \forall \mathtt{x}\, \psi \,\&\, \varphi \Leftrightarrow \forall \mathtt{x}\, (\psi \,\&\, \varphi)$;

в) $\models \varphi \vee \exists \mathtt{x}\, \psi \Leftrightarrow \exists \mathtt{x}\, (\varphi \vee \psi)$ $u \models \exists \mathtt{x}\, \psi \vee \varphi \Leftrightarrow \exists \mathtt{x}\, (\psi \vee \varphi)$;

г) $\models \varphi \vee \forall \mathtt{x}\, \psi \Leftrightarrow \forall \mathtt{x}\, (\varphi \vee \psi)$ $u \models \forall \mathtt{x}\, \psi \vee \varphi \Leftrightarrow \forall \mathtt{x}\, (\psi \vee \varphi)$. **(клас)**

Доказателство. **(а)** Да изберем произволни структура $\mathbf{M}$ и оценка v в $\mathbf{M}$. Нека A е съждението $\mathbf{M} \models \varphi[v]$ и за произволен елемент μ на универсума на $\mathbf{M}$, нека $B(\mu)$ е съждението $\mathbf{M} \models \psi[\mathtt{x} := \mu|v]$. Тъй като $\mathtt{x}$ не е свободна променлива на φ, то за всеки елемнт μ на универсума на $\mathbf{M}$, съждението $\mathbf{M} \models \varphi[\mathtt{x} := \mu|v]$ е вярно тогава и само тогава, когато е вярно съждението A (което очевидно не зависи от μ).

Най-напред ще докажем, че $\mathbf{M} \models \varphi \,\&\, \exists \mathtt{x}\, \psi \Rightarrow \exists \mathtt{x}\, (\varphi \,\&\, \psi)[v]$. Да допуснем, че съждението A е вярно и съществува такова $\mu \in |\mathbf{M}|$, че съждението $B(\mu)$ е вярно. В такъв случай очевидно съществува такова $\mu \in |\mathbf{M}|$, че съждението „A и $B(\mu)$“ е вярно.

За да докажем обратната посока, да допуснем, че съществува такова $\mu \in |\mathbf{M}|$, че съждението „A и $B(\mu)$“ е вярно. В такъв случай очевидно съждението A ще бъде вярно и освен това ще съществува $\mu \in |\mathbf{M}|$, за което съждението $B(\mu)$ е вярно.

(б) Да изберем произволни структура $\mathbf{M}$ и оценка v в $\mathbf{M}$. Нека A е съждението $\mathbf{M} \models \varphi[v]$ и за произволен елемент μ на универсума на $\mathbf{M}$, нека $B(\mu)$ е съждението $\mathbf{M} \models \psi[\mathtt{x} := \mu|v]$. Тъй като $\mathtt{x}$ не е свободна променлива на φ, то за всеки елемнт μ на универсума на $\mathbf{M}$, съждението $\mathbf{M} \models \varphi[\mathtt{x} := \mu|v]$ е вярно тогава и само тогава, когато е вярно съждението A (което очевидно не зависи от μ).

Най-напред ще докажем, че $\mathbf{M} \models \varphi \,\&\, \forall \mathtt{x}\, \psi \Rightarrow \forall \mathtt{x}\, (\varphi \,\&\, \psi)[v]$. Да допуснем, че е вярно A и че за всяко $\mu \in |\mathbf{M}|$ е вярно $B(\mu)$. В такъв случай очевидно за всяко $\mu \in |\mathbf{M}|$ ще бъде вярно съждението „A и $B(\mu)$“.

За да докажем обратната посока, да допуснем, че за всяко $\mu \in |\mathbf{M}|$ е вярно съждението „A и $B(\mu)$“. Тъй като универсумът на $\mathbf{M}$ е непразен, като приложим току-що допуснатото за някой елемент на универсума, ще получим, че съждението A е вярно. Освен това очевидно за всяко $\mu \in |\mathbf{M}|$ ще бъде вярно съждението $B(\mu)$.

да разберем дали е вярно A, или за всяко $\mu \in |\mathbf{M}|$ е вярно $B(\mu)$. Това ни подсказва, че съждението не е вярно конструктивно и значи трябва да допуснем противното.

И така, да допуснем, че не е вярно съждението „A е вярно или за всяко $\mu \in |\mathbf{M}|$ е вярно $B(\mu)$“. Това означава (вж. доказателството на съждение 3.7.1 а)), че съждението A не е вярно и съждението „за всяко $\mu \in |\mathbf{M}|$ е вярно $B(\mu)$“ също не е вярно. Но ние знаем, че за всяко $\mu \in |\mathbf{M}|$ е вярно „A или $B(\mu)$“. Тъй като току-що установихме, че A не е вярно, то значи за всяко $\mu \in |\mathbf{M}|$ е вярно $B(\mu)$. Това е противоречие. ■

3.7.11. Твърдение. *За произволни формули φ и ψ, ако $\mathtt{x}$ не е свободна променлива на φ, то:*

а) $\models (\varphi \Rightarrow \forall \mathtt{x}\, \psi) \Leftrightarrow \forall \mathtt{x}\, (\varphi \Rightarrow \psi)$;

б) $\models (\varphi \Rightarrow \exists \mathtt{x}\, \psi) \Leftrightarrow \exists \mathtt{x}\, (\varphi \Rightarrow \psi)$; **(клас)**

в) $\models (\exists \mathtt{x}\, \psi \Rightarrow \varphi) \Leftrightarrow \forall \mathtt{x}\, (\psi \Rightarrow \varphi)$;

г) $\models (\forall \mathtt{x}\, \psi \Rightarrow \varphi) \Leftrightarrow \exists \mathtt{x}\, (\psi \Rightarrow \varphi)$. **(клас)**

✓ **3.7.12. Дефиниция.** Казваме, че една формула е в *пренексна нормална форма*, ако тя има вида

$$\mathtt{Я}_1 \mathtt{x}_1 \mathtt{Я}_2 \mathtt{x}_2 \ldots \mathtt{Я}_n \mathtt{x}_n (\varphi)$$

където $\mathtt{Я}_1, \mathtt{Я}_2, \ldots, \mathtt{Я}_n$ са квантори ($\forall$ или $\exists$), а формулата φ не съдържа квантори.

✓ **3.7.13. Твърдение.** *За всяка формула може да се намери еквивалентна на нея (според класическата логика) формула, която е в пренексна нормална форма.*

✓ Доказателство. Нека ни е дадена произволна формула. Съгласно лема 2.6.11 може да намерим конгруентна, а значи и еквивалентна на нея формула, в която всички квантори имат различни променливи и никоя кванторна променлива не е свободна променлива във формулата. Да прилагаме в така получената формула докато може замени от

следния вид:

$$\begin{gathered}
\varphi \,\&\, \exists \mathtt{x}\, \psi \longmapsto \exists \mathtt{x}\, (\varphi \,\&\, \psi) \\
\exists \mathtt{x}\, \psi \,\&\, \varphi \longmapsto \exists \mathtt{x}\, (\psi \,\&\, \varphi) \\
\varphi \,\&\, \forall \mathtt{x}\, \psi \longmapsto \forall \mathtt{x}\, (\varphi \,\&\, \psi) \\
\forall \mathtt{x}\, \psi \,\&\, \varphi \longmapsto \forall \mathtt{x}\, (\psi \,\&\, \varphi) \\
\varphi \vee \exists \mathtt{x}\, \psi \longmapsto \exists \mathtt{x}\, (\varphi \vee \psi) \\
\exists \mathtt{x}\, \psi \vee \varphi \longmapsto \exists \mathtt{x}\, (\psi \vee \varphi) \\
\varphi \vee \forall \mathtt{x}\, \psi \longmapsto \forall \mathtt{x}\, (\varphi \vee \psi) \\
\forall \mathtt{x}\, \psi \vee \varphi \longmapsto \forall \mathtt{x}\, (\psi \vee \varphi) \\
\varphi \Rightarrow \forall \mathtt{x}\, \psi \longmapsto \forall \mathtt{x}\, (\varphi \Rightarrow \psi) \\
\varphi \Rightarrow \exists \mathtt{x}\, \psi \longmapsto \exists \mathtt{x}\, (\varphi \Rightarrow \psi)
\end{gathered}$$

$$\begin{gathered}
\exists \mathtt{x}\, \psi \Rightarrow \varphi \longmapsto \forall \mathtt{x}\, (\psi \Rightarrow \varphi) \\
\forall \mathtt{x}\, \psi \Rightarrow \varphi \longmapsto \exists \mathtt{x}\, (\psi \Rightarrow \varphi) \\
\neg \exists \mathtt{x}\, \varphi \longmapsto \forall \mathtt{x}\, \neg \varphi \\
\neg \forall \mathtt{x}\, \varphi \longmapsto \exists \mathtt{x}\, \neg \varphi
\end{gathered}$$

Ако имаме подформула от вида напр. $\varphi \,\&\, \exists \mathtt{x}\, \psi$, то във φ променливата $\mathtt{x}$ няма да бъде свободна, защото ако $\mathtt{x}$ бе свободна във φ, то тъй като работим с формула, в която няма два квантора с една и съща променлива, то променливата $\mathtt{x}$ би била свободна и в цялата формула, а пък си осигурихме да няма кванторна променлива, която да е свободна в цялата формула. Това означава, че съгласно твърдения 3.7.10 б), 3.7.10 а), 3.7.10 г), 3.7.10 в), 3.7.11 а), 3.7.11 б), 3.7.11 в), 3.7.11 г), 3.7.2 а) и 3.7.2 б) при всяка една от тези замени ще получаваме еквивалентни формули. Ако след краен брой стъпки стигнем формула, към която не може да прилагаме никоя от по-горните замени, то значи сме получили формула в пренексна нормална форма. Това ни дава частичната коректност на алгоритъма за привеждане в пренексна нормална форма. Остава да докажем, че алгоритъмът винаги спира.

(I начин) За произволна формула χ да наречем *дълбочина* на подформулата χ' на χ дължината на най-дългата редица от вида

$$\chi_1, \chi_2, \chi_3, \ldots, \chi_k$$

където $\chi_1 = \chi$, $\chi_k = \chi'$, $\chi_i \neq \chi_{i+1}$, χ_{i+1} е подформула на χ_i и формулите $\chi_1, \chi_2, \ldots, \chi_{k-1}$ не започват с квантор. Ако си мислим формулата като

дърво, тогава дълбочината на една подформула е равна на разстоянието между корена на поддървото и корена на цялото дърво, като при това пропускаме възлите от дървото, в които стои квантор.

Да забележим, че замените от горния вид не променят броя на подформулите, започващи с квантор. Всяка една такава замяна обаче намалява дълбочината на някоя от подформулите, започващи с квантор, а това не може да продължи до безкрайност.

(II начин) Да разгледаме следния начин, по който от формула получаваме аритметичен израз. Да заменим всяка атомарна подформула с числото 2. Да заменим всяка подформула от вида $\chi' \,\&\, \chi'$, $\chi' \vee \chi''$ и $\chi' \Rightarrow \chi''$ с $x + y$, където x и y са изразите, получени съответно от χ' и χ''. Да заменим всяка подформула от вида $\neg\chi$ с $1 + x$, където x е изразът, получен от χ. Да заменим всяка подформула от вида $\forall \mathtt{x}\, \chi$ и $\exists \mathtt{x}\, \chi$ с x^2, където x е изразът, получен от χ. Може да се забележи, че така полученият израз е естествено число, не по-малко от 2. Тъй като за всеки естествени числа, не по-малки от 2, е изпълнено $x + y^2 < (x + y)^2$, $x^2 + y < (x + y)^2$ и $1 + x^2 < (1 + x)^2$, то след всяко преобразование на формулата по описание по-горе начин, стойността на съответния аритметичен израз ще се увеличи. Това обаче не може да продължи до безкрай, защото броят на символите в тези аритметични изрази се запазва след всяко такова преобразование и значи може да се получат само краен брой аритметични изрази. ∎

3.7.14. ***Забележка*:** Въпреки че правилата за замяна в алгоритъма за привеждане в пренексна нормална форма са много на брой, те се помнят лесно. Нека да забележим, че когато кванторът излиза пред отрицание или се е намирал отляво на импликация, той се променя, т.е. от $\forall$ става $\exists$ и от $\exists$ става $\forall$. Във всички останали случаи кванторът се премества без промяна.

✓ **3.7.15. Твърдение.** *За всяка формула може да се намери еквивалентна на нея (според класическата логика) формула, която е едновременно в пренексна нормална форма и в отрицателна нормална форма.*

✓ Доказателство. Непосредствено се проверява, че ако преобразуваме според алгоритъма за привеждане в пренексна нормална форма формула, която се намира в отрицателна нормална форма, то ще получаваме формули в отрицателна нормална форма. Това означава, че можем просто да приведем формулата в отрицателна нормална фор-

ма и след това да приложим алгоритъмът за привеждане в пренексна нормална форма.

Също така можем непосредствено да проверим, че ако преобразуваме според алгоритъма за привеждане в отрицателна нормална форма формула, която се намира в пренексна нормална форма, то ще получаваме формули в пренексна нормална форма. Това означава, че можем просто да приведем формулата в пренексна нормална форма и след това да приложим алгоритъмът за привеждане в отрицателна нормална форма.

Също така, напълно допустимо е да прилагаме разбъркано замените от двата алгоритъма. В този случай процесът също със сигурност ще бъде краен и ще даде желания резултат, но тук няма да доказваме това. ∎

3.7.16. ***Забележка*:** Ако формулата вече е приведена в отрицателна нормална форма и приложим към нея алгоритъма за привеждане в пренексна нормална форма, няма да ни се наложи да прилагаме нито веднъж замени, при които кванторът се променя, т.е. от $\forall$ става $\exists$ или от $\exists$ става $\forall$. Това означава, че не е нужно да работим стъпка по стъпка, а можем просто да преместим наведнъж всички квантори отпред на формулата (обаче без да променяме реда им!) и ще получим пренексна нормална форма.

✓ **3.7.17. Твърдение.** *Формула от вида* $\forall \mathtt{x}\,(\varphi)$ *е тъждествено вярна в някоя структура* $\mathbf{M}$ *тогава и само тогава, когато в* $\mathbf{M}$ *е тъждествено вярна формулата* φ.

✓ Доказателство. Да допуснем, че $\mathbf{M} \vDash \forall \mathtt{x}\,(\varphi)$, т.е. за всяка оценка v в $\mathbf{M}$ е вярно $\mathbf{M} \vDash \forall \mathtt{x}\,(\varphi)[v]$. По дефиниция това означава, че за всяка оценка v в $\mathbf{M}$ и за всеки елемент μ на универсума на $\mathbf{M}$ е вярно $\mathbf{M} \vDash \varphi[\mathtt{x} := \mu|v]$. В частност, ако изберем $\mu = v(\mathtt{x})$, оценката $[\mathtt{x} := \mu|v]$ ще съвпадне с v и значи за всяка оценка v в $\mathbf{M}$ ще бъде вярно $\mathbf{M} \vDash \varphi[v]$. Следователно φ е тъждествено вярна в $\mathbf{M}$.

Обратната посока се вижда още по-лесно. Да допуснем, че φ е тъждествено вярна в $\mathbf{M}$. Това значи, че φ ще бъде вярна при произволна оценка. В частност φ ще бъде вярна и при всяка модифицирана оценка $[\mathtt{x} := \mu|v]$, и значи формулата $\forall \mathtt{x}\,\varphi$ е тъждествено вярна в $\mathbf{M}$. ∎

Прилагателното „скулемов“ от следващата дефиниция произлиза от името на откривателя на преобразованието, наречено *скулемизация* — норвежкия логик Тӯралф Скӯлем.

При скулемизацията ще използваме субституции, които променят една единствена променлива. За удобство ще използваме следното означение (подобно на означението, което имаме за оценки): ако $\mathtt{x}$ е променлива, а τ — терм, то с $\mathtt{x} \coloneqq \tau$ ще означаваме субституцията, която заменя $\mathtt{x}$ с τ и оставя всички останали променливи непроменени.

✓ **3.7.18. Дефиниция.** а) Нека е дадена формула от вида $\exists \mathtt{x}\,(\varphi)$, в която няма свободни променливи, а $\mathtt{c}$ е символ за константа. Формулата $\varphi[\mathtt{x} \coloneqq \mathtt{c}]$ се нарича *скулемово усилване* (от първи вид) на $\exists \mathtt{x}\,(\varphi)$.

б) Нека е дадена формула от вида $\exists \mathtt{x}\,(\varphi)$, чиито свободни променливи са $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$, а $\mathtt{f}$ е n-местен функционален символ. Формулата $\varphi[\mathtt{x} \coloneqq \mathtt{f}(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)]$ се нарича *скулемово усилване* (от втори вид) на $\exists \mathtt{x}\,(\varphi)$.

в) Символът за константа $\mathtt{c}$ от а) и функционалният символ $\mathtt{f}$ от б) се наричат *скулемов символ за константа* и *скулемов функционален символ*.

3.7.19. Пример. Да разгледаме формулата $\varphi = \exists \mathtt{x}\,\mathtt{p}(\mathtt{x})$. Едно нейно скулемово усилване е формулата $\varphi' = \mathtt{p}(\mathtt{c})$. Във всяка структура $\mathbf{M}$ формулата φ „казва“, че в универсума на $\mathbf{M}$ има елемент, за който предикатът $\mathtt{p}^{\mathbf{M}}$ е истина. Формулата φ' пък казва, че предикатът $\mathtt{p}^{\mathbf{M}}$ е истина не за някой неопределен елемент, а за $\mathtt{c}^{\mathbf{M}}$. Следователно скулемовото усилване φ' казва повече, отколкото φ и ако формулата φ' е вярна в някоя структура, то и φ ще бъде вярна. По-долу твърдение 3.7.21 ще покаже, че това се случва винаги, а не само при този конкретен пример.

Ако имаме право сами да решим каква да бъде интерпретацията на символа $\mathtt{c}$ и формулата φ е вярна в $\mathbf{M}$, то ще можем да интерпретираме $\mathtt{c}$ така, че $\mathtt{c}^{\mathbf{M}}$ да бъде оня елемент от универсума, чието съществуване се твърди от формулата φ. Ако променим $\mathbf{M}$ по този начин, формулата φ' ще стана вярна. По-долу твърдение 3.7.22 ще покаже, че това се случва винаги при скулемово усилване от първи вид, а не само при този конкретен пример.

3.7.20. Пример. Да разгледаме формулата $\varphi = \exists \mathtt{y}\,\mathtt{p}(\mathtt{x}, \mathtt{y})$. Едно нейно скулемово усилване е формулата $\varphi' = \mathtt{p}(\mathtt{x}, \mathtt{f}(\mathtt{x}))$. Във всяка структура $\mathbf{M}$ формулата φ „казва“, че за всеки елемент μ на универсума на $\mathbf{M}$ съществува такъв елемент ν, че $\mathtt{p}^{\mathbf{M}}(\mu, \nu)$ е истина. Формулата φ' пък казва, че $\mathtt{p}^{\mathbf{M}}(\mu, \nu)$ е истина не за някое неопределено ν, а за $\nu = \mathtt{f}^{\mathbf{M}}(\mu)$. Следователно скулемовото усилване φ' казва повече, отколкото φ и ако

формулата φ' е вярна в някоя структура, то и φ ще бъде вярна. По-долу твърдение 3.7.21 ще покаже, че това се случва винаги, а не само при този конкретен пример.

Ако имаме право сами да решим каква да бъде интерпретацията на символа $\mathtt{f}$ и формулата φ е вярна в $\mathbf{M}$, то ще можем да интерпретираме $\mathtt{f}$ така, че $\mathtt{f}^{\mathbf{M}}(\mu)$ да бъде оня елемент ν от универсума, чието съществуване се твърди от формулата φ. Ако променим $\mathbf{M}$ по този начин, формулата φ' ще стана вярна. По-долу твърдение 3.7.22 ще покаже, че това се случва винаги при скулемово усилване от втори вид, а не само при този конкретен пример.

Да припомним твърдение 3.3.17, съгласно което за произволни формула φ със свободни променливи $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$, термове $\tau_1, \tau_2, \ldots, \tau_n$ и оценка v в структура $\mathbf{M}$

$$\mathbf{M} \vDash \varphi[\tau_1, \ldots, \tau_n][v] \longleftrightarrow \mathbf{M} \vDash \varphi[[\![\tau_1]\!]^{\mathbf{M}} v, \ldots, [\![\tau_n]\!]^{\mathbf{M}} v] \tag{17}$$

Следващото твърдение обяснява защо скулемовото усилване е наречено „усилване“ — скулемовото усилване на една формула винаги казва повече неща, отколкото самата формула.

✓ **3.7.21. Твърдение.** *Ако скулемовото усилване на една формула е тъждествено вярно в структура* $\mathbf{M}$*, то и самата формула е тъждествено вярна в* $\mathbf{M}$*.*

✓ Доказателство. Нека $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ са свободните променливи на $\exists \mathtt{x}\,(\varphi)$; тогава свободните променливи на φ са измежду $\mathtt{x}, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$.

Нека $\varphi[\mathtt{x}, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n]$ е скулемово усилване на $\exists \mathtt{x}\,(\varphi)$, което е тъждествено вярно в $\mathbf{M}$. За да докажем $\mathbf{M} \vDash \exists \mathtt{x}\,(\varphi)$, да изберем произволна оценка v в $\mathbf{M}$. Трябва да докажем $\mathbf{M} \vDash \exists \mathtt{x}\,(\varphi)[v]$. От $\mathbf{M} \vDash \varphi[\mathtt{x}, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n]$ следва

$$\mathbf{M} \vDash \varphi[\mathtt{x}, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n][v]$$

Съгласно твърдение 3.3.17 това е еквивалентно на

$$\mathbf{M} \vDash \varphi[\mathtt{x}, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq [\![\tau]\!]^{\mathbf{M}} v, v(\mathtt{x}_1), v(\mathtt{x}_2), \ldots, v(\mathtt{x}_n)]$$

което съгласно дефиниция 3.3.2 ж) ни дава

$$\mathbf{M} \vDash \exists \mathtt{x}\,(\varphi)[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq v(\mathtt{x}_1), v(\mathtt{x}_2), \ldots, v(\mathtt{x}_n)]$$

т.е.

$$\mathbf{M} \vDash \exists \mathtt{x}\,(\varphi)[v]$$

■

✓ **3.7.22. Твърдение.** *Нека φs е скулемово усилване на $\exists \mathtt{x}\,(\varphi)$ и скулемовият символ не се среща никъде във формулата φ. Ако $\mathbf{M} \vDash \exists \mathtt{x}\,(\varphi)$, то съществува структура $\mathbf{K}$, която съвпада с $\mathbf{M}$ във всичко, освен може би при интерпретацията на скулемовия символ, такава, че $\mathbf{K} \vDash \varphi s$.*

Доказателство. Доказателството ще извършим по отделно в зависимост от това дали скулемовото усилване е от първи или втори вид.

(първи вид) Съгласно дефиниция 3.7.18 субституцията s има вида $\mathtt{x} \coloneqq \mathtt{c}$, където символът за константа $\mathtt{c}$ не се среща никъде във формулата φ, а формулата $\exists \mathtt{x}\,(\varphi)$ няма свободни променливи. Тъй като $\mathbf{M} \vDash \exists \mathtt{x}\,(\varphi)$, то съгласно дефиниция 3.3.2 ж) получаваме, че съществува такова $\mu \in |\mathbf{M}|$, че

$$\mathbf{M} \vDash \varphi[\mathtt{x} \coloneqq \mu]$$

Нека структурата $\mathbf{K}$ е идентична с $\mathbf{M}$ във всичко, освен в интерпретацията на символа $\mathtt{c}$ — нека $\mathtt{c}^{\mathbf{K}} = \mu$.

За да установим, че $\mathbf{K} \vDash \varphi[\mathtt{x} \coloneqq \mathtt{c}]$, да изберем произволна оценка v. Трябва да докажем

$$\mathbf{M} \vDash \varphi[\mathtt{x} \coloneqq \mathtt{c}][v]$$

Съгласно твърдение 3.3.17 това е еквивалентно на

$$\mathbf{M} \vDash \varphi[\mathtt{x} \coloneqq [\![\mathtt{c}]\!]^{\mathbf{M}} v]$$

т.е. на

$$\mathbf{M} \vDash \varphi[\mathtt{x} \coloneqq \mu]$$

което вече видяхме, че е вярно.

(втори вид) Съгласно дефиниция 3.7.18 субституцията s има вида $\mathtt{x} \coloneqq \mathtt{f}(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)$, където функционалният символ $\mathtt{f}$ не се среща никъде във формулата φ, а $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ са свободните променливи на $\exists \mathtt{x}\,(\varphi)$. Тъй като $\mathbf{M} \vDash \exists \mathtt{x}\,(\varphi)$, то за произволни $\mu_1, \mu_2, \ldots, \mu_n \in |\mathbf{M}|$

$$\mathbf{M} \vDash \exists \mathtt{x}\,(\varphi)[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \mu_1, \mu_2, \ldots, \mu_n]$$

откъдето съгласно дефиниция 3.3.2 ж) получаваме, че за произволни $\mu_1, \mu_2, \ldots, \mu_n \in |\mathbf{M}|$ съществува такова $\mu \in |\mathbf{M}|$, че

$$\mathbf{M} \vDash \varphi[\mathtt{x}, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \mu, \mu_1, \mu_2, \ldots, \mu_n] \tag{18}$$

Нека $f\colon |\mathbf{M}|^n \to |\mathbf{M}|$ е функция, която на произволно избрани $\mu_1, \mu_2, \ldots, \mu_n$ съпоставя така намереното μ.*

Нека структурата $\mathbf{K}$ е идентична с $\mathbf{M}$ във всичко, освен в интерпретацията на символа $\mathtt{f}$ — нека $\mathtt{f}^{\mathbf{K}} = f$.

За да установим, че $\mathbf{K} \vDash \varphi[\mathtt{x} \coloneqq \mathtt{f}(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n)]$, да изберем произволна оценка v. Трябва да докажем

$$\mathbf{M} \vDash \varphi[\mathtt{x}, \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \mathtt{f}(\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n), \mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n][v]$$

Съгласно твърдение 3.3.17 това е еквивалентно на

$$\mathbf{M} \vDash \varphi[\mathtt{x}, \mathtt{x}_1, \ldots, \mathtt{x}_n \coloneqq [\![\mathtt{f}(\mathtt{x}_1, \ldots, \mathtt{x}_n)]\!]^{\mathbf{M}} v, [\![\mathtt{x}_1]\!]^{\mathbf{M}} v, \ldots, [\![\mathtt{x}_n]\!]^{\mathbf{M}} v]$$

т.е. на

$$\mathbf{M} \vDash \varphi[\mathtt{x}, \mathtt{x}_1, \ldots, \mathtt{x}_n \coloneqq f(v(\mathtt{x}_1), \ldots, v(\mathtt{x}_n)), v(\mathtt{x}_1), \ldots, v(\mathtt{x}_n)]$$

Последното е истина предвид (18) и дефиницията на функцията f. ∎

Забележка: В доказателството на твърдение 3.7.22 за втория вид скулемово усилване използвахме т. н. *аксиома за избора*. Един начин да формулираме тази аксиома е следният:

> Ако за всяко $x \in X$ съществува $y \in Y$, за които е верен някакъв предикат $p(x, y)$, то тогава съществува такава функция f, че за всяко $x \in X$ е вярно $p(x, f(x))$.

Ако тук интерпретираме квантора „съществува“ конструктивно, тази аксиома е съвсем естествена, защото в този случай ще разполагаме с конкретен метод, посредством който по дадено x можем да получим нужното y. Когато интерпретираме квантора класически, не разполагаме с никакъв метод, посредством който по x можем да получим y. Затова аксиомата за избора дълго време е била една от най-оспорваните аксиоми на теория на множествата. Всъщност методът на скулемизацията, може да се обоснове и без да се използва аксиомата за избора, но доказателството става значително по-усложнено.

✓ **3.7.23. Дефиниция.** Множество от формули е *изпълнимо*, ако съществува структура, в която са тъждествено верни всички формули от множеството. Множество от формули е *неизпълнимо*, ако не е изпълнимо.

*На това място използваме т. н. *аксиома за избора*.

3.7.24. (скулемизация) Нека ни е дадено крайно множество от формули Γ в пренексна нормална форма и да разгледаме следния процес, който ще наречем *скулемизация*. Избираме произволна формула от Γ, която съдържа квантор. Ако формулата започва с квантор $\forall$, то отстраняваме този квантор. Съгласно твърдение 3.7.17 ще получим такова множество от формули Γ', че Γ е изпълнимо тогава и само тогава, когато Γ е изпълнимо. Ако пък избраната формула започва с квантор $\exists$, то да заменим тази формула с такова нейно скулемово усилване, че СКУЛЕМОВИЯТ ФУНКЦИОНАЛЕН СИМВОЛ ДА НЕ СЕ СРЕЩА В НИКОЯ ФОРМУЛА ОТ Γ.* Съгласно твърдение 3.7.21, ако Γ' е изпълнимо, то и Γ ще е изпълнимо. Ще докажем и обратното — ако Γ е изпълнимо, то и Γ' е изпълнимо. Да допуснем, че Γ е изпълнимо и нека $\mathbf{M}$ е някоя структура, в която са тъждествено верни формулите от Γ. Съгласно твърдение 3.7.22 или 3.7.22 съществува структура $\mathbf{K}$, в която е вярно скулемовото усилване на формулата, започваща с $\exists$. Останалите формули са идентични в Γ и Γ'. Те обаче не съдържат скулемовия символ, а структурите $\mathbf{M}$ и $\mathbf{K}$ съвпадат за всички символи, освен евентуално за скулемовия. Тъй като тези формули са тъждествено верни в $\mathbf{M}$, то те са тъждествено верни и в $\mathbf{K}$.

Тъй като на всяка стъпка от описания процес изчезва по един квантор, то след краен брой стъпки ще получим множество, в което всички формули са безквантворни. При това, така намереното множество не се различава от първоначалното по отношение на своята изпълнимост — ако първоначалното множество е изпълнимо, то и новополученото ще е неизпълнимо, и ако новополученото е изпълнимо, то значи и първоначалното множество е било изпълнимо.

И така, доказахме следната теорема:

✓ **3.7.25. Теорема.** *Нека Γ е крайно** множество от формули. Ако в сигнатурата има достатъчно символи, то ще можем да намерим такова крайно множество Γ' от безквантворни формули, че Γ да бъде изпълнимо тогава и само тогава, когато е изпълнимо Γ'.*

Доказателство. Да приведем формулите от Γ в пренексен вид и след това да приложим скулемизация (вж. 3.7.24). ∎

✓ **3.7.26. Дефиниция.** Казваме, че една формула е в *скулемова нормална форма*, ако:

*Разбира се, може да направим това само ако в сигнатурата има достатъчно символи.

**Теоремата е вярна и за безкрайно Γ, но тук няма да обосноваваме това.

– формулата е в пренексна нормална форма;

– формулата не съдържа нито един квантор $\exists$;

– формулата не съдържа свободни променливи.

✓ **3.7.27. Теорема за скулемизацията.** *Нека* Γ *е крайно*[*] *множество от формули. Ако в сигнатурата има достатъчно символи, то ще можем да намерим такова крайно множество* Γ' *от формули в скулемова нормална форма, че* Γ *да бъде изпълнимо тогава и само тогава, когато е изпълнимо* Γ'.

✓ Доказателство. Най-напред, използвайки теорема 3.7.25 можем да получим крайно множество Δ от безкванторни формули, което е изпълнимо тогава и само тогава, когато е изпълнимо Γ. Ако пред всяка от безкванторните формули в Δ сложим достатъчно квантори за всеобщност, ще получим множество Γ' в скулемова нормална форма. Освен това, съгласно твърдение 3.7.17, формулите от множеството Γ' са тъждествено верни точно в онези структури, в които са тъждествено верни и формулите от Δ. Следователно Δ е изпълнимо тогава и само тогава, когато е изпълнимо Γ'. ∎

3.7.28. Твърдение. *За всеки три формули* φ, ψ *и* χ*:*

а) $\vDash \varphi \vee (\psi \,\&\, \chi) \Leftrightarrow (\varphi \vee \psi) \,\&\, (\varphi \vee \chi)$

б) $\vDash (\psi \,\&\, \chi) \vee \varphi \Leftrightarrow (\psi \vee \varphi) \,\&\, (\chi \vee \varphi)$

Доказателство. Да изберем произволна структура $\mathbf{M}$ и оценка v в $\mathbf{M}$. Нека A е съждението $\mathbf{M} \vDash \varphi[v]$, B е съждението $\mathbf{M} \vDash \psi[v]$ и C е съждението $\mathbf{M} \vDash \chi[v]$. Трябва да докажем, че са верни твърденията

> „Съждението (A или (B и C)) е вярно тогава и само тогава, когато е вярно ((A или B) и (A или C))“

и

> „Съждението ((B и C) или A) е вярно тогава и само тогава, когато е вярно ((B или A) и (C или A))“

И двете твърдения са очевидни. ∎

✓ **3.7.29. Дефиниция.**
Литерал означава атомарна формула или отрицание на атомарна формула.

[*] Теоремата е вярна и за безкрайно Γ, но тук няма да обосноваваме това.

б) *Елементарна дизюнкция* означава безкванторна формула, в която единствените логически операции са дизюнкция и отрицание и всяко отрицание се намира пред атомарна формула.

в) Една безкванторна формула е в *конюнктивна нормална форма*, ако представлява конюнкция от елементарни дизюнкции.*

✓ **3.7.30. Твърдение.** *За всяка безкванторна формула можем да намерим еквивалентна на нея (според класическата логика) формула, която е в конюнктивна нормална форма.*

✓ Доказателство. Нека φ е произволна безкванторна формула. Да я приведем в отрицателна нормална форма.** По този начин ще получим безкванторна формула, в която единствените логически операции са конюнкция, дизюнкция и отрицания и всяко отрицание се намира пред атомарна формула. Да прилагаме към така получената формула докато може замени от следния вид:

$$\varphi \vee (\psi \,\&\, \chi) \longmapsto (\varphi \vee \psi) \,\&\, (\varphi \vee \chi) \tag{19}$$

$$(\psi \,\&\, \chi) \vee \varphi \longmapsto (\psi \vee \varphi) \,\&\, (\chi \vee \varphi) \tag{20}$$

Съгласно твърдение 3.7.28 при замени от този вид ще получаваме еквивалентни формули. Освен това след всяка такава замяна формулата ще остава безкванторна, няма да се появят други операции, освен конюнкция, дизюнкция и отрицание и всяко отрицание ще си остане пред атомарната си формула. Може да забележим обаче, че ако след краен брой стъпки стигнем до формула, към която не можем да приложим

*По точно, можем да дадем следната индуктивна дефиниция на това какво значи *конюнктивна нормална форма*:

а) всяка елементарна дизюнкция е в конюнктивна нормална форма;

б) конюнкция на две формули в конюнктивна нормална форма е формула в конюнктивна нормална форма.

**Да припомним, че за целта е достатъчно да прилагаме докато може замени от следния вид:

$$\varphi \Leftrightarrow \psi \longmapsto (\varphi \Rightarrow \psi) \,\&\, (\psi \Rightarrow \varphi)$$

$$\varphi \Rightarrow \psi \longmapsto \neg\varphi \vee \psi$$

$$\neg\neg\varphi \longmapsto \varphi$$

$$\neg(\varphi \vee \psi) \longmapsto \neg\varphi \,\&\, \neg\psi$$

$$\neg(\psi \,\&\, \varphi) \longmapsto \neg\psi \vee \neg\varphi$$

замяна от горния вид, това ще означава, че сме получили формула в конюнктивна нормална форма. Това показва частичната коректност на алгоритъма за привеждане в конюнктивна нормална форма. Остава да видим защо алгоритъмът винаги свършва след краен брой стъпки.

(I начин) За произволна формула χ да наречем *дълбочина* на подформулата χ' на χ дължината на най-дългата редица от вида

$$\chi_1, \chi_2, \chi_3, \ldots, \chi_k$$

където $\chi_1 = \chi$, $\chi_k = \chi'$, $\chi_i \neq \chi_{i+1}$, χ_{i+1} е подформула на χ_i и формулите $\chi_1, \chi_2, \ldots, \chi_{k-1}$ не са от вида $\chi' \,\&\, \chi''$. Ако си мислим формулата като дърво, тогава дълбочината на една подформула е равна на разстоянието между корена на поддървото и корена на цялото дърво, като при това пропускаме възлите от дървото, в които стои конюнкция.

Може да се забележи, че ако формулата ψ се получава от φ посредством някоя от замените (19) и (20), то броят на подформулите от вида $\chi' \,\&\, \chi''$ не се променя, а дълбочината на някоя подформула от този вид намалява. Това не може да продължи до безкрайност.

(II начин) Да разгледаме следния начин, по който от безкванторна формула в отрицателна нормална форма получаваме аритметичен израз. Да заменим всеки литерал с числото 2. Да заменим всяка подформула от вида $\chi' \,\&\, \chi'$ с $x + y$, където x и y са изразите, получени съответно от χ' и χ''. Да заменим всяка подформула от вида $\chi' \vee \chi'$ с $x.y$, където x и y са изразите, получени съответно от χ' и χ''. След всяко преобразование на формулата от горния вид, стойността на съответния аритметичен израз няма да се промени, защото $x.(y + z) = x.y + x.z$ и $(y + z).x = y.x + z.x$. Същевременно след всяко такова преобразование се увеличава броят на числата 2 в така получения аритметичен израз. Това не може да продължи до безкрайност, защото за числа не по-малки от 2 е вярно $x + y \leqq x.y$ и значи стойността на всеки такъв аритметичен израз със сигурност е не по-малка от удвоения брой на числата 2 в него. ∎

3.7.31. Твърдение. *Ако в сигнатурата разполагаме с достатъчно неизползвани символи за константи и функционални символи,[*] то за всяко крайно множество[**] от формули можем да намерим такова*

[*] На практика това не е сериозно ограничение, защото при нужда винаги можем да заменим сигнатурата с нова сигнатура, в която има много нови символи.

[**] Това твърдение е вярно и за безкрайни множества, но тук няма да доказваме това.

крайно множество от елементарни дизюнкции, че (от гледна точка на класическата логика) първоначалното множество е изпълнимо тогава и само тогава, когато е изпълнимо множеството от елементарни дизюнкции.

Доказателство. Съгласно теорема 3.7.25 за всяко крайно множество от формули можем да намерим крайно множество от безкванторни формули, което е изпълнимо тогава и само тогава, когато е изпълнимо първоначалнорагато множество. Също така видяхме, че за всяка безкванторна формула можем да намерим еквивалентна на нея безкванторна формула в конюнктивна нормална форма.

Всяка формула от вида $\varphi \& \psi$ е тъждествено вярна в някоя структура $\mathbf{M}$ тогава и само тогава, когато формулата $\varphi \& \psi$ е вярна при всяка оценка в $\mathbf{M}$, а това е така тогава и само тогава, когато при всяка оценка в $\mathbf{M}$ са верни двете формули φ и ψ, и значи тогава и само тогава, когато тези две формули са тъждествено верни в $\mathbf{M}$. Това означава, че ако в така полученото множество заменяме докато може всяка формула от вида $\varphi \& \psi$ с двете формули φ и ψ, ще получаваме формули, които са изпълними тогава и само тогава, когато е изпълнимо първоначалното множеството от формули. Тъй като след всяка такава замяна броят на конюнкциите намалява, то след краен брой стъпки ще стигнем до множество от елементарни дизюнкции. ■

Твърдение 3.7.31 показва, че когато се интересуваме от изпълнимостта на множество от формули по отношение на класическата логика, може да считаме, че елементите на това множество имат сравнително прост вид — елементарни дизюнкции.

изобразяват така намерения извод, използвайки дървоподобна* структура, изглеждаща по следния начин:

$$\dfrac{\dfrac{\mathrm{C} \quad \mathrm{O_2}}{\mathrm{CO_2}} \qquad \dfrac{\mathrm{MgO} \quad \mathrm{H_2}}{\mathrm{H_2O}}}{\mathrm{H_2CO_3}}$$

Всяка от хоризонталните черти в този запис отговаря на прилагането на някое от правилата.

***Задача* 46:** Дадени са правилата

$$\frac{}{\mathtt{p}} \qquad \frac{\mathtt{p} \quad \mathtt{p}}{\mathtt{q}} \qquad \frac{\mathtt{r}}{\mathtt{q}} \qquad \frac{\mathtt{s} \quad \mathtt{q}}{\mathtt{r}} \qquad \frac{\mathtt{q} \quad \mathtt{p}}{\mathtt{s}}$$

Използвайки обратна изводимост, изведете `r`.

Представане на правила посредством предикатни формули

За да можем да се възползваме наготово от всички неща, които сме направили за предикатната логика, е удобно да представяме правилата като специален вид формули, наречени хорнови клаузи.

✓ **4.1.1. Дефиниция.** Нека $n \geqq 0$ и формулите $\varphi_1, \varphi_2, \ldots, \varphi_n, \psi$ са атомарни. *Хорнови клаузи* ще наричаме формулите от следните два вида:

– ако $n = 0$, то атомарната формула ψ е хорнова клауза;

– ако $n \geqq 1$, то формулата $\varphi_1 \,\&\, \varphi_2 \,\&\, \ldots \,\&\, \varphi_n \Rightarrow \psi$ е хорнова клауза.

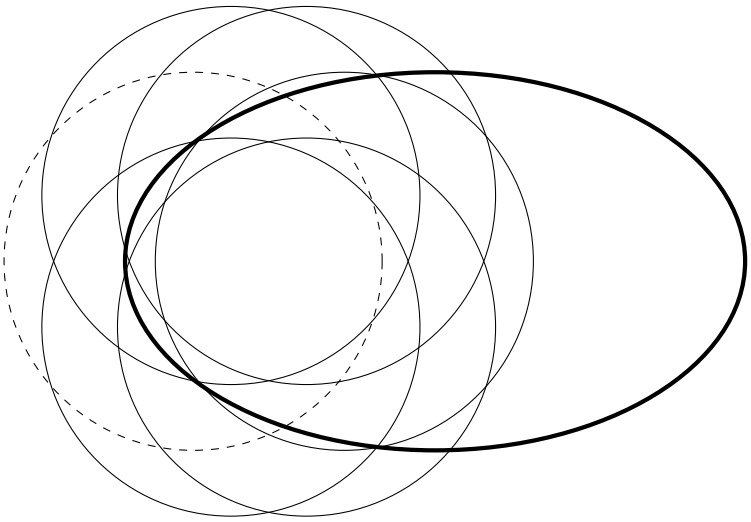
Атомарните формули $\varphi_1, \varphi_2, \ldots, \varphi_n$ се наричат *предпоставки* на клаузата, а атомарната формулата ψ — нейно *заключение*. За горните клаузи (включително когато $n = 0$) ще използваме следния запис:

$$\psi \texttt{ :- } \varphi_1, \varphi_2, \ldots, \varphi_n$$

Клаузите без предпоставки (т. е. когато $n = 0$) се наричат *факти*. За краткост ще изпускаме прилагателното „хорнова“ и ще казваме само *клауза*.

Забележка: Въпреки че на пролог фактите се записват без използването на знака `:-`, за да е ясно, че става въпрос за клауза, тук ще пишем ψ `:-` дори когато клаузата е без предпоставки. В контекста на пролог думата „правило“ означава не това, което бе казано по-горе. В езика пролог „правило“ означава хорнова клауза с поне една предпоставка. Следователно ако използваме терминологията на пролог, може да кажем, че клаузите са два вида: факти и правила.

*Логиците са единствените математици, при които дърветата растат в правилната посока, т.е. нагоре, а не надолу.



Фиг. 13. Диаграма на Вен за вярна клауза

4.1.2. Следствие. *Клаузата* ψ :- $\varphi_1, \varphi_2, \ldots, \varphi_n$ *е тъждествено вярна в структура* $\mathbf{M}$ *тогава и само тогава, когато при всяка оценка в* $\mathbf{M}$*, при която предпоставките* $\varphi_1, \varphi_2, \ldots, \varphi_n$ *са верни, заключението* ψ *също е вярно.*

Доказателство. Уверете се сами, че това наистина е така, използвайки дефиницията за тъждествена вярност (3.4.1 а)) и дефиницията за формула вярна в структура при оценка (3.3.2). Убедете се, че твърдението е вярно независимо дали $n = 0$ или $n \geqq 1$.

Обърнете внимание, че в общия случай за всяка предпоставка може да има оценки, при които тя е вярна, и оценки, при които тя не е вярна. Също и за заключението може да има оценки, при които то е вярно, и оценки, при които то не е вярно. А за да бъде една клауза тъждествено вярна в структура, ние искаме следното — при онези оценки, при които се окаже, че всички предпоставки са верни, заключението също да бъде вярно. Ако за някоя оценка дори една от предпоставките се окаже невярна, за такава оценка клаузата не казва нищо. Това е илюстрирано във фиг. 13. Всяка от окръжностите изобразява множеството от оценки, при които някоя от предпоставките е вярна. Елипсата с дебела линия пък е множеството от оценки, при които заключението е вярно. Вътрешната бяла област са оценките, при които всяка от предпоставките се оказва вярна и тази бяла област се включва изцяло в удебелената елипса. Забележете, че ако обръщаме внимание не на всички предпоставки, а само на една (напр. на щрихованата окръжност),

тогава в общия случай няма да открием никаква зависимост между нея и заключението. Зависимост съществува само когато вземем предвид всички предпоставки едновременно. ■

✓ **4.1.3. Дефиниция.** Нека $n \geqq 0$ и формулите $\varphi_1, \varphi_2, \ldots, \varphi_n$ са атомарни. *Запитвания* се наричат формулите от следните два вида:

- когато $n = 0$, то формулата $\top$ е запитване (да припомним, че $\top = \neg\bot$ е винаги вярна формула);
- ако $n \geqq 1$, то формулата $\varphi_1 \,\&\, \varphi_2 \,\&\, \ldots \,\&\, \varphi_n$ е запитване

За запитванията (включително когато $n = 0$) ще използваме следния запис:

$$\texttt{?-}\ \varphi_1, \varphi_2, \ldots, \varphi_n$$

Запитването, което не съдържа нито една атомарна формула (т. е. когато $n = 0$), се нарича *празно запитване*.

4.1.4. Следствие. *а) Запитването* $\texttt{?-}\ \varphi_1, \varphi_2, \ldots, \varphi_n$ *е изпълнимо в структура* $\mathbf{M}$ *тогава и само тогава, когато може да се намери оценка в* $\mathbf{M}$*, при която формулите* $\varphi_1, \varphi_2, \ldots, \varphi_n$ *са верни.*

б) Празното запитване `?-` *е вярно във всяка структура.*

4.2. Права изводимост с частни случаи

Понятие за изводимост с частни случаи

Предпоставките и заключенията от правилата в примера за химичен синтез, който използвахме, за да се запознаем с правата и обратната изводимост (вж. стр. 215), не съдържаха променливи. Това, оказва се, спестява невероятно много усложнения. В този раздел ще разгледаме по-формално изводимостта с без променливи. Разбира се, вместо изрази като H_2O и NaCl ще използваме атомарни формули.

Нека например универсумът, с който работим, се състои от естествените числа. Да разгледаме клаузата:

$$\texttt{p(x, x, y) :-}$$

Някои от нещата, които се използват в нашата програма, са дефинирани не в самата нея, а се вземат наготово или от някоя програмна библиотека, или пък са вградени в самия език за програмиране. Следователно нещата, които се дефинират в нашата програма, разширяват съвкупност от вече съществуващи неща, които не е нужно ние да дефинираме. За такъв тип разширения в математическата логика се използва терминът „обогатяване“.

Да дефинираме формално какво означава обогатяване на сигнатура и обогатяване на структура:

4.2.2. Дефиниция. а) Сигнатурата $\mathfrak{sig}_1$ е *обогатяване* на $\mathfrak{sig}_2$, ако всеки символ на $\mathfrak{sig}_2$ е символ и на $\mathfrak{sig}_1$.

б) Структурата $\mathbf{M}_1$ е *обогатяване* на структурата $\mathbf{M}_2$, ако двете структури имат един и същи универсум, сигнатурата на $\mathbf{M}_1$ е обогатяване на сигнатурата на $\mathbf{M}_2$ и символите от сигнатурата на $\mathbf{M}_2$ се интерпретират по един и същи начин в двете структури.

Горната дефиниция показва, че ако $\mathbf{M}_1$ е обогатяване на $\mathbf{M}_2$, то всичко, което можем да кажем в $\mathbf{M}_2$, можем да го кажем и в $\mathbf{M}_1$, обаче в структурата $\mathbf{M}_1$ може би можем да казваме и неща, които не могат да се кажат в $\mathbf{M}_2$. Следващото твърдение формализира това наблюдение.

4.2.3. Твърдение. *Нека структурата* $\mathbf{K}$ *е обогатяване на* $\mathbf{M}$ *и* v *е оценка в* $\mathbf{M}$ *(което, разбира се, означава, че* v *е оценка и в* $\mathbf{K}$*). Тогава всеки терм* τ *от сигнатурата на* $\mathbf{M}$ *и всяка формула* φ *от сигнатурата на* $\mathbf{M}$ *са също терм и формула от сигнатурата на* $\mathbf{K}$ *и освен това:*

$$[\![\tau]\!]^{\mathbf{M}} v = [\![\tau]\!]^{\mathbf{K}} v$$
$$\mathbf{M} \vDash \varphi[v] \longleftrightarrow \mathbf{K} \vDash \varphi[v]$$

Доказателство. С тривиална индукция по τ и φ, използвайки това че според дефиницията за обогатяване символите от τ и φ се интерпретират по един и същ начин в двете структури. ∎

✓ **4.2.4.** Да фиксираме сигнатура ВГРАД, която ще съдържа символите, чийто смисъл не се определя от програмиста, а са вградени в езика за програмиране или идват наготово от някоя програмна библиотека. За символите от сигнатурата ВГРАД ще казваме, че са *вградени*. Да

(обджектив си), а структурата — *реализация* или *тяло*.

фиксираме също и структура $\mathbf{B}$, която дава интерпретацията на вградените символи. Когато използваме език за програмиране, ние дефинираме свои собствени функции и предикати, но не променяме смисъла на вградените функции и предикати. Следователно нашата програма дефинира структура, която е обогатяване на $\mathbf{B}$. Когато използваме функционален език за програмиране, това обогатяване ще включва нови функции, а когато използваме логически език за програмиране — нови предикати.

Тъй като искаме да разработим теорията на логическите езици, в които се дефинират само нови предикати, ще обърнем по-специално внимание на този тип обогатявания.

4.2.5. Дефиниция. Когато всички нови символи в дадена обогатена сигнатура или структура са предикатни символи (т.е. не са добавени символи за константи или функционални символи), то тогава такава сигнатура или структура ще наричаме *предикатно обогатяване*.

Псевдоформули

В началото на този раздел казахме, че когато универсумът се състои например от естествените числа, то клауза като $\mathtt{p}(\mathtt{x},\mathtt{x},\mathtt{y})\,\mathtt{:-}$ е еквивалентна на съвкупност от частни случаи от вида $\mathtt{p}(0,0,0)\,\mathtt{:-}$, $\mathtt{p}(0,0,1)\,\mathtt{:-}$, $\mathtt{p}(0,0,2)\,\mathtt{:-}$ и т.н. Обаче тези частни случаи не са клаузи, защото всяка клауза представлява редица от символи, докато използваните тук частни случаи са изрази, включващи елементи на универсума на структурата, в конкретния случай естествени числа. Изрази като $\mathtt{p}(0,0,1)$ дори не са формули. Ще нарачем такива изрази „псевдоформули“.

4.2.6. Дефиниция. *Псевдоформула* при сигнатура $\mathfrak{sig}$ е израз от вида $\mathtt{p}(\beta_1,\beta_2,\ldots,\beta_n)$, където $\mathtt{p}$ е n-местен предикатен символ от сигнатурата $\mathfrak{sig}$, а $\beta_1,\beta_2,\ldots,\beta_n$ са елементи на универсума на $\mathbf{B}$.

Забележка: По принцип би трябвало обектите от горната дефиниция да бъдат наречени „атомарни псевдоформули без функционални символи“. Но тъй като няма да ни трябват по-сложни псевдоформули, ще си позволим за краткост да казваме просто „псевдоформули“.

Верността на псевдоформулите може да се дефинира по очаквания начин:

4.2.7. Дефиниция. Нека структурата $\mathbf{M}$ е обогатяване на $\mathbf{B}$. Ако $\mathtt{p}(\beta_1,\beta_2,\ldots,\beta_n)$ е псевдоформула при сигнатурата на $\mathbf{M}$, то ще казваме,

Сега бихме могли да дефинираме и какво означава една псевдоклауза да бъде вярна в структура, но няма да ни се налага да правим това.

Формална дефиниция на правата изводимост с частни случаи

4.2.11. Дефиниция. За едно множество от клаузи Γ ще казваме, че е *логическа програма*, ако всичките му клаузи са от сигнатура, която е предикатно обогатяване на ВГРАД, и отляво на символа :- не се съдържат вградени предикатни символи (т.е предикатни символи от ВГРАД).

Да си припомним, че при правата изводимост извеждахме новите факти на стъпки. Ако Γ е логическа програма и X са нещата, които сме доказали до момента, то с $T_\Gamma(X)$ ще означим нещата, които можем да докажем на една стъпка от X посредством Γ.

✓ **4.2.12. Дефиниция.** а) Нека Γ е логическа програма. Ако псевдоклаузата ψ :- $\varphi_1, \varphi_2, \ldots, \varphi_n$ е частен случай на елемент на Γ и всяка от псевдоформулите $\varphi_1, \varphi_2, \ldots, \varphi_n$ е или елемент на множеството X, или е вярна в структурата $\mathbf{B}$, то казваме, че ψ се *извежда едностъпково с частни случаи* от X при програма Γ.

б) Ако Γ е логическа програма, а X — множество от псевдоформули, да означим с $T_\Gamma(X)$ множеството от всички псевдоформули, които се извеждат едностъпково от X при програма Γ.

Една особеност на горната дефиниция, която не бе илюстрирана от неформалния пример за химичен синтез (стр. 215), е това че не е нужно да извеждаме псевдоформулите, използващи вградени предикатни символи. За такива псевдоформули директно проверяваме дали са верни в $\mathbf{B}$.

4.2.13. Пример. Нека универсум на структурата $\mathbf{B}$ са естествените числа и предикатният символ = се интерпретира като равенство. Нека Γ е логическа програма и някоя клауза от Γ има частен случай

$$\mathtt{p}(5) \ \texttt{:-}\ 7 = 7, \mathtt{p}(2), 1 = 1, \mathtt{p}(3)$$

Дори множеството Δ да съдържа само $\mathtt{p}(2)$ и $\mathtt{p}(3)$, то от Δ при програма Γ ще се извежда едностъпково с частни случаи псевдоформулата $\mathtt{p}(5)$.

Следващото твърдение показва, че операторът T_Γ е монотонен:

✓ **4.2.14. Твърдение.** *Ако $X \subseteq Y$, то $T_\Gamma(X) \subseteq T_\Gamma(Y)$.*

✓ Доказателство. Нека ψ е произволен елемент на $T_\Gamma(X)$. Тогава Γ съдържа клауза ψ :- $\varphi_1, \varphi_2, \ldots, \varphi_n$, чиито предпоставки $\varphi_1, \varphi_2, \ldots, \varphi_n$ са елементи на X или верни в $\mathbf{B}$. Но $X \subseteq Y$, следователно всяка от предпоставките е или елементи на Y, или вярна в $\mathbf{B}$. Следователно ψ се извежда едностъпково и от Y, т. е. $\psi \in T_\Gamma(Y)$. ∎

Нека Γ е множеството от клаузите, с които работим. Да си представим как бихме извеждали нови факти. Ако на стъпка n имаме изведени атомарните формули от множеството X, то на стъпка $n+1$ към тези формули се добавят и формулите от $T_\Gamma(X)$. Следователно формулите, които имаме на стъпка $n+1$, са $X \cup T_\Gamma(X)$. Нека в началото (на стъпка 0) множеството от фактите, които сме извели, е празно:

$$\varnothing$$

След една стъпка множеството от факти, които знаем, става

$$\varnothing \cup T_\Gamma(\varnothing)$$

След още една стъпка към тези факти се добавят още и фактите от $T_\Gamma(\varnothing \cup T_\Gamma(\varnothing))$, следователно новото множество от изведени факти е:

$$\varnothing \cup T_\Gamma(\varnothing) \cup T_\Gamma(\varnothing \cup T_\Gamma(\varnothing))$$

Аналогично, след още една стъпка получаваме

$$\varnothing \cup T_\Gamma(\varnothing) \cup T_\Gamma(\varnothing \cup T_\Gamma(\varnothing)) \cup T_\Gamma(\varnothing \cup T_\Gamma(\varnothing) \cup T_\Gamma(\varnothing \cup T_\Gamma(\varnothing)))$$

И т. н. Ако горните изрази ви изглеждат сложни и все по-сложни, то усещането не е само ваше. Дали не би било по-добре да работим не с $T_\Gamma(X)$, а с $T'_\Gamma(X) = X \cup T_\Gamma(X)$? Ако вместо T_Γ използваме T'_Γ, то горните сметки дават много по-прости изрази. В началото (на стъпка 0) множеството от фактите, които сме извели, е празно:

$$\varnothing$$

След една стъпка множеството от факти, които знаем, става

$$T'_\Gamma(\varnothing)$$

След още една стъпка получаваме

$$T'_\Gamma(T'_\Gamma(\varnothing))$$

Аналогично, след още една стъпка имаме

$$T'_\Gamma(T'_\Gamma(T'_\Gamma(\varnothing)))$$

И т. н. Очевидно при T'_Γ получаваме много по-прости изрази. Това, което може би не е очевидно, е следното: не е нужно да използваме T'_Γ вместо T_Γ, защото горните изрази, използващи T'_Γ, ще си запазят стойността, ако в тях вместо T'_Γ използваме T_Γ. Например

$$T'_\Gamma(T'_\Gamma(T'_\Gamma(\varnothing))) = T_\Gamma(T_\Gamma(T_\Gamma(\varnothing)))$$

Причината за това е следната. Операторът T_Γ е монотонен, т. е. ако $X \subseteq Y$, то $T_\Gamma(X) \subseteq T_\Gamma(Y)$. Ясно е че

$$\varnothing \subseteq T_\Gamma(\varnothing)$$

От тук монотонността на T_Γ ни дава

$$T_\Gamma(\varnothing) \subseteq T_\Gamma(T_\Gamma(\varnothing))$$

След това пак заради монотонността получаваме

$$T_\Gamma(T_\Gamma(\varnothing)) \subseteq T_\Gamma(T_\Gamma(T_\Gamma(\varnothing)))$$

Значи имаме следната верига

$$\varnothing \subseteq T_\Gamma(\varnothing) \subseteq T_\Gamma(T_\Gamma(\varnothing)) \subseteq T_\Gamma(T_\Gamma(T_\Gamma(\varnothing))) \subseteq \ldots$$

Да, но от $X \subseteq T_\Gamma(X)$ следва $X \cup T_\Gamma(X) = T_\Gamma(X)$, т. е. $T'_\Gamma(X) = T_\Gamma(X)$. Следователно спокойно можем да работим с T_Γ вместо с T'_Γ.

Видяхме, че $T^n_\Gamma(\varnothing) \subseteq T^{n+1}_\Gamma(\varnothing)$, като $T^n_\Gamma(\varnothing)$ съдържа атомарните формули, които можем да докажем с не повече от n-стъпково доказателство. Значи ако означим

$$T^\infty_\Gamma(\varnothing) = \varnothing \cup T_\Gamma(\varnothing) \cup T_\Gamma(T_\Gamma(\varnothing)) \cup T_\Gamma(T_\Gamma(T_\Gamma(\varnothing))) \cup \ldots$$

то $T^\infty_\Gamma(\varnothing)$ ще бъде множеството от всички неща, които може да се докажат (все едно с колко дълго доказателство).

✓ **4.2.15. Дефиниция.** а) Да положим $T^\infty_\Gamma(\varnothing) = \bigcup_{i=0}^\infty T^i_\Gamma(\varnothing)$.

б) Казваме, че φ се извежда с права изводимост с частни случаи при програма Γ, ако $\varphi \in T^\infty_\Gamma(\varnothing)$.

Казахме, че $T_\Gamma^\infty(\varnothing)$ съдържа нещата, които могат да се докажат с произволно дълго (но все пак крайно) доказателство. А дали ако допуснем за верни нещата от $T_\Gamma^\infty(\varnothing)$, от тях няма да може да докажем нещо ново? Ако да, то това би означавало, че някои неща не могат да се докажат с крайно доказателство.

За щастие това не се случва. По-точно, ако приложим оператора T_Γ към всички неща, които имат крайно доказателство, то няма да получим нищо ново. Значи

$$T_\Gamma^\infty(\varnothing) = T_\Gamma(T_\Gamma^\infty(\varnothing))$$

Причината горното равенство да е вярно е свойство на оператора T_Γ, което се нарича компактност — за да изведем кой да е конкретен елемент на $T_\Gamma(X)$, са ни нужни само краен брой елементи на X. Ако операторът T_Γ не беше компактен,* то редицата

$$\varnothing \subseteq T_\Gamma(\varnothing) \subseteq T_\Gamma(T_\Gamma(\varnothing)) \subseteq T_\Gamma(T_\Gamma(T_\Gamma(\varnothing))) \subseteq \dots$$

би могла да се окаже трансфинитна и бихме могли да я продължим по следния начин:

$$\begin{gathered}\varnothing \subseteq T_\Gamma(\varnothing) \subseteq T_\Gamma(T_\Gamma(\varnothing)) \subseteq T_\Gamma(T_\Gamma(T_\Gamma(\varnothing))) \subseteq \dots \subseteq \\ \subseteq T_\Gamma^\omega(\varnothing) \subseteq T_\Gamma(T_\Gamma^\omega(\varnothing)) \subseteq T_\Gamma(T_\Gamma(T_\Gamma^\omega(\varnothing))) \subseteq T_\Gamma(T_\Gamma(T_\Gamma(T_\Gamma^\omega(\varnothing)))) \subseteq \dots \\ \subseteq T_\Gamma^{\omega.2}(\varnothing) \subseteq T_\Gamma(T_\Gamma^{\omega.2}(\varnothing)) \subseteq T_\Gamma(T_\Gamma(T_\Gamma^{\omega.2}(\varnothing))) \subseteq T_\Gamma(T_\Gamma(T_\Gamma(T_\Gamma^{\omega.2}(\varnothing)))) \subseteq \dots \\ \subseteq \dots \subseteq \dots \subseteq \dots\end{gathered}$$

където

$$\begin{aligned}T_\Gamma^\omega(\varnothing) &= \varnothing \cup T_\Gamma(\varnothing) \cup T_\Gamma(T_\Gamma(\varnothing)) \cup T_\Gamma(T_\Gamma(T_\Gamma(\varnothing))) \cup \dots \\ T_\Gamma^{\omega.2}(\varnothing) &= T_\Gamma^\omega(\varnothing) \cup T_\Gamma(T_\Gamma^\omega(\varnothing)) \cup T_\Gamma(T_\Gamma(T_\Gamma^\omega(\varnothing))) \cup \dots \\ &\dots\end{aligned}$$

✓ **4.2.16. Твърдение.** *а)* $T_\Gamma(T_\Gamma^\infty(\varnothing)) \subseteq T_\Gamma^\infty(\varnothing)$;

б) ако $T_\Gamma(X) \subseteq X$, *то* $T_\Gamma^\infty(\varnothing) \subseteq X$;

в) $T_\Gamma(T_\Gamma^\infty(\varnothing)) = T_\Gamma^\infty(\varnothing)$.

*В математическата логика се разглеждат и изводимости, при които правилата имат безброй много предпоставки. При такива изводимости операторът T_Γ не е компактен.

✓ Доказателство. **(а)** Да допуснем, че $\psi \in T_\Gamma(T_\Gamma^\infty(\varnothing))$. Тогава ψ се извежда едностъпково от $T_\Gamma^\infty(\varnothing)$ при програма Г и значи Г съдържа такава клауза ψ :- $\varphi_1, \varphi_2, \ldots, \varphi_n$, че формулите $\varphi_1, \varphi_2, \ldots, \varphi_n$ са от $T_\Gamma^\infty(\varnothing)$. По дефиниция

$$T_\Gamma^\infty(\varnothing) = \bigcup_{i=0}^{\infty} T_\Gamma^i(\varnothing)$$

следователно за всяко $i \in \{1, 2, \ldots, n\}$ съществува такова число k_i, че $\varphi_i \in T_\Gamma^{k_i}(\varnothing)$. Нека $m = \max\{k_1, k_2, \ldots, k_n\}$. Тъй като множествата $T_\Gamma^i(\varnothing)$ се включват едно в друго, то $T_\Gamma^{k_i}(\varnothing) \subseteq T_\Gamma^m(\varnothing)$ за всяко $i \in \{1, 2, \ldots, n\}$ и значи $\varphi_i \in T_\Gamma^m(\varnothing)$, от където следва $\psi \in T_\Gamma^{m+1}(\varnothing) \subseteq T_\Gamma^\infty(\varnothing)$.

(б) Ясно е, че

$$\varnothing \subseteq X$$

От тук монотонността на T_Γ ни дава

$$T_\Gamma(\varnothing) \subseteq T_\Gamma(X) \subseteq X$$

От $T_\Gamma(\varnothing) \subseteq X$ пак от монотонността получаваме

$$T_\Gamma(T_\Gamma(\varnothing)) \subseteq T_\Gamma(X) \subseteq X$$

Продължавайки по същия начин получаваме $T_\Gamma^i(\varnothing) \subseteq X$ за произволно i. Следователно $T_\Gamma^\infty(\varnothing) \subseteq X$, тъй като $T_\Gamma^\infty(\varnothing)$ е обединение на всички множества от вида $T_\Gamma^i(\varnothing)$.

(в) От $T_\Gamma(T_\Gamma^\infty(\varnothing)) \subseteq T_\Gamma^\infty(\varnothing)$ монотонността ни дава

$$T_\Gamma(T_\Gamma(T_\Gamma^\infty(\varnothing))) \subseteq T_\Gamma(T_\Gamma^\infty(\varnothing))$$

Следователно може да приложим б) при $X = T_\Gamma(T_\Gamma^\infty(\varnothing))$. Получаваме $T_\Gamma^\infty(\varnothing) \subseteq T_\Gamma(T_\Gamma^\infty(\varnothing))$, което заедно с а) ни дава исканото. ∎

***Задача* 47:** Нека сигнатурата ВГРАД не съдържа нито един предикатен символ и частните случаи на елементите на Г могат да се генерират алгоритмично. Намерете алгоритъм, който генерира елементите на $T_\Gamma^\infty(\varnothing)$.

4.2.17. Задача 47 показва, че в някои случаи е възможно да генерираме алгоритмично елементите на $T_\Gamma^\infty(X)$. Всъщност оказва се, че на практика много често това наистина е възможно. Следователно ако φ се извежда с права изводимост с частни случаи от Г, то има начин

това да се установи алгоритмично — просто трябва да си генерираме търпеливо елементите на $T_\Gamma^\infty(\varnothing)$, чакайки да се появи φ. Ако φ се извежда с права изводимост с частни случаи от Г, то рано или късно* ще попаднем на φ. Ако обаче φ не се извежда с права изводимост с частни случаи от Г, тогава този алгоритъм ще генерира до безкрайност нови и нови елементи на $T_\Gamma^\infty(\varnothing)$ с надеждата да получи φ, т. е. ще се зацикли. За алгоритъм, имащ тези свойства, казваме, че *полурешава* задачата дали някоя атомарна формула се извежда с права изводимост с частни случаи от Г. Този дефект за съжаление е неотстраним — в теория на изчислимостта се доказва, че обикновено не съществува алгоритъм, който *решава* тази задача и никога не се зацикля.

Понятие за коректност и пълнота

В предния подраздел дефинирахме какво значи една псевдоформула да се извежда при програма Г. Но откъде знаем, че това, което сме дефинирали, работи както трябва? Всъщност отникъде. Дали няма много неща, които следват от клаузите в Г, но не могат да се докажат по начина, използван в предния подраздел? Или може би се случва нещо още по-лошо — с дефинирания метод се извеждат абсурдни неща, които очевидно няма как да са верни? Можем ли да спим спокойно при това положение?

Начинът за възстановяване на спокойствието е следният: първо ще дефинираме какво всъщност искаме. А след това ще проверим математически съответствието между това, което искаме, и това, което имаме.

А какво искаме? Искаме когато някоя псевдоформула се извежда, това да означава, че тя е вярна (в някакъв, все още неясно какъв смисъл). За момента имаме дефиниция какво означава псевдоформула да бъде вярна в структура. Но в коя структура трябва да бъде вярна изведената псевдоформула? Програмата Г не уточнява това. Тя само казва, че някакви клаузи трябва да се верни, но те може да бъдат верни в много структури. Може би тогава трябва да поискаме изведените неща да бъдат верни в структурите, които са модел на програмата Г? Да, това ще ни свърши работа, но със следното уточнение: интересуваме се от онези структури, които не само са модел на Г, но освен това са и обогатявания на **B**. И така, време е да дадем следната дефиниция:

✓ **4.2.18. Дефиниция.** Нека сигнатурата $\mathfrak{sig}$ е предикатно обогатяване на ВГРАД и Г е логическа програма при сигнатура $\mathfrak{sig}$. Ако φ е псевдоформула при сигнатура $\mathfrak{sig}$, която не използва вграден предикатен

*По-скоро късно, отколкото рано.

символ, ще казваме че тя *се удовлетворява* при програма Γ, ако φ е вярна във всички структури за $\mathfrak{sig}$, които са едновременно предикатно обогатяване на $\mathbf{B}$ и модел на Γ.

Понятието „удовлетворява“ е важно, но очевидно то не е директно проверимо от компютър. Структурите, при които клаузите в Γ се оказват верни, могат да бъдат най-разнообразни — толкова разнообразни, че не само компютър, ами и човек не може да ги опише всичките. Как тогава компютър ще може да провери дали във всяка такава структура е вярна псевдоформулата φ? А дори и да се окаже възможно да опишем всички структури, в които клаузите от Γ са верни, те най-често ще бъдат безброй много и няма да може с компютър да проверим дали φ е вярна във всяка една от тези безброй много структури. Но да допуснем, че по някакви причини се окаже възможно да се ограничим само с една структура. Ако универсумът ѝ е безкраен, в общия случай задачата пак става нерешима![*]

От друга страна, в предния подраздел дефинирахме и понятието „φ се извежда с права изводимост с частни случаи при програма Γ“, което може да се полурешава алгоритмично. Изобщо от никъде не личи каква е връзката между двете понятия „φ се удовлетворява при програма Γ“ и „φ се извежда с права изводимост с частни случаи при програма Γ“. Дефинициите на тези две понятия са толкова различни една от друга! И въпреки това, оказва се, че те са еквивалентни.

✓ **4.2.19.** Когато някоя изводимост е такава, че всяко нещо, което може да се докаже е вярно, тогава казваме, че тази изводимост е **коректна**. А когато всяко вярно нещо може да се докаже, тогава казваме, че изводимостта е **пълна**.

Например когато кажем, че правата изводимост с частни случаи е коректна, това означава, че ако φ се извежда с права изводимост с частни случаи при програма Γ, то φ се удовлетворява при програма Γ. А когато кажем, че тя е пълна, това означава, че ако φ се удовлетворява при програма Γ, то φ се извежда с права изводимост с частни случаи при програма Γ.

Теоремата за коректност и пълнота на дадена изводимост е дълбок резултат, от изключителна важност в математическата логика, кой-

[*] Например ако универсумът е $\mathbb{N}\backslash\{0\}$, функционалният символ „+“ се интерпретира като събиране, ^ като степенуване и 2 като числото две, то атомарната формула $\mathtt{x}_1\mathtt{^}(\mathtt{y}_1 + 2) + \mathtt{x}_2\mathtt{^}(\mathtt{y}_2 + 2) = \mathtt{x}_3\mathtt{^}(\mathtt{y}_3 + 2)$ е изпълнима тогава и само тогава, когато великата теорема на Ферма е невярна. Няма как да искаме от компютрите да решат задача, която на хората им е отнела векове!

то се доказва в почти всеки учебник по логика. Ако погледнем по-философски на тази теорема, тя казва, че едно нещо е доказуемо тогава и само тогава, когато то е вярно. Доказуемост и вярност са две напълно различни понятия и тяхната еквивалентност е нещо удивително!

Да разберем, че нещо е доказуемо, означава да разполагаме с доказателство, което сме в състояние да проверим стъпка по стъпка и което е достатъчно подробно, така че да можем да осъществим проверката изцяло механично, без да се замисляме за каквото и да е. Но и след най-акуратната проверка на едно такова доказателство ние така и няма да разберем защо всъщност доказаното нещо е вярно. От друга страна, да разберем защо нещо е вярно означава да се сдобием с интуиция за това как всъщност стоят нещата, която да ни убеди, че нещото наистина е вярно. Но колкото и добра да е интуицията на един математик, винаги има случаи, когато тя го заблуждава, така че единственият сигурен начин да се убедим във верността на нещо е да имаме негово доказателство. По този начин виждаме, че всяко едно от тези две неща — използването на доказателствата като окончателно и безапелативно свидетелство за вярност и придобиването на интуиция, която „от пръв поглед“ да ни казва как стоят нещата, кое е вярно и кое не — е важно и необходимо в математиката.

Интересно е това, че за тези две неща се грижат напълно различни дялове от главния мозък. Светът на лявото полукълбо е подреден свят, където за всичко си има точни правила. Всяко нещо в този свят кристално ясно и съществува необвързано и само по себе си. За речта — говорима или писмена — отговаря лявото полукълбо. От друга страна, светът на дясното полукълбо е объркан. Всичко в този свят е свързано с всичко и се влияе от всичко. В този свят няма „да“ и „не“, няма „вярно“ и „невярно“, а едно цялостно усещане за съотношението на всяко нещо с всички останали неща. Дясното полукълбо е нямо и вместо реч прави музика.*

Математиката е може би най-точната наука и затова изглежда естествено за нея да отговаря изцяло лявото полукълбо на мозъка. Оказва се обаче, че това съвсем не е така. За формулирането на логически правилни разсъждения наистина отговаря лявата половина на мозъка, но математическата интуиция и математическите идеи са отдясно. Новата математика се твори от дясната половина на мозъка, а се контролира и проверява от лявата.

*Всъщност дясното полукълбо не познава обикновения смисъл на думите, но познава преносния им смисъл и пише стихотворения. Повече подробности за функциите на двете полукълба на мозъка може да се научат напр. от [13].

Ако четем и заучаваме някое точно математическо доказателство, но интуицията и идеите, скрити в него, ни убягват, това ще бъде безполезно. И също така да четем или слушаме нечии математически идеи, които не са съпроводени с точни доказателства, е все едно да слушаме нечии празни фантазии — на такива фантазии никой математик няма да обърне сериозно внимание (и с право!).

Когато студентите учат някой математически предмет (напр. логическо програмиране), те винаги трябва да търсят скритите идеи и интуицията, обясняващи смисъла дефинициите и изясняващи доказателствата. Също така те трябва да изучават как могат да изказват тези идеи на точен математически език. И когато това стане, те ще могат да се движат свободно в абстрактния свят на математиката и да творят, знаейки, че винаги ще бъдат в състояние да облекат измислените неща в точен математически формализъм, който да им гарантира, че не са измислили някоя глупост и с който ще могат да споделят идеите си. Ако студентите учат нещата наизуст, без всъщност да ги разбират, те най-вероятно ще си вземат изпита, но неприятните усещания, които са изпитвали докато учат, могат да предизвикат у един съпричастен преподавател единствено съчувствие и съжаление. Студенти пък, които са се опитвали да учат идеите без да проследяват доказателствата, са направили нещо безполезно, което е може би по-подходящо за философски факултет, отколкото за ФМИ. Обаче студенти, които са успели да схванат истински нещата, са радост за преподавателите, защото такива студенти са успели поне донякъде да видят красотите в чудния, хем измислен, хем истински свят, в който преподавателите са се опитали да ги заведат.

Забележка: Можем накратко да резюмираме казаното до тук само с едно изречение: „Използвайте целия си мозък, а не само едната му половина!“

Положителна диаграма на структура

В дефиницията на понятието „φ се удовлетворява при програма Γ“ удовлетворяването на φ се установява посредством верността на φ в определени структури. В дефиницията на понятието „φ се извежда с права изводимост с частни случаи при програма Γ“ пък, извеждането на φ се установява посредством принадлежността на φ на определено множество от псевдоформули. Следователно ако искаме да свържем тези две понятия, имаме нужда от връзка между структурите и множествата от псевдоформули.

(в) Нека $\mathbf{K}$ е предикатно обогатяване на $\mathbf{B}$ и модел на Γ. Съгласно лема 4.2.22, $T_\Gamma(\mathrm{diag}(\mathbf{K})) \subseteq \mathrm{diag}(\mathbf{K})$. Според твърдение 4.2.16 б) това означава, че $\mathrm{diag}(\mathbf{M}) = T_\Gamma^\infty(\varnothing) \subseteq \mathrm{diag}(\mathbf{K})$. ■

Теореми за коректност и пълнота на правата изводимост с частни случаи

Почти всички доказателства за коректност в математическата логика се доказват по следния начин — първо показваме, че след едностъпково доказателство от верни неща получаваме пак верни неща. Щом след една стъпка, ще получим верни неща, значи след още една пак ще получим верни неща и т. н. С други думи, всички неща, които могат да се изведат, са верни и значи имаме коректност.

В следващото доказателство твърдението, че след едностъпково доказателство от верни неща получаваме пак верни неща, изглежда така: $T_\Gamma(\mathrm{diag}(\mathbf{M})) \subseteq \mathrm{diag}(\mathbf{M})$. Твърдението пък, че всички неща, които могат да се изведат, са верни, изглежда така: $T_\Gamma^\infty(\varnothing) \subseteq \mathrm{diag}(\mathbf{M})$.

4.2.25. Теорема за коректност на правата изводимост с частни случаи. *Нека* Γ *е логическа програма. Ако някоя псевдоформула се извежда с права изводимост с частни случаи при програма* Γ*, то тя се удовлетворява при програма* Γ*.*

Доказателство. Нека φ се извежда с права изводимост с частни случаи при програма Γ. По дефиниция това означава, че $\varphi \in T_\Gamma^\infty(\varnothing)$ и значи φ не използва вграден предикатен символ.

Трябва да докажем, че ако $\mathbf{M}$ е структура за сигнатурата на Γ, която е едновременно обогатяване на $\mathbf{B}$ и модел на Γ, то $\mathbf{M}$ е модел на φ. Но щом $\mathbf{M}$ е модел на Γ, то от лема 4.2.22 получаваме, че $T_\Gamma(\mathrm{diag}(\mathbf{M})) \subseteq \mathrm{diag}(\mathbf{M})$, и значи твърдение 4.2.16 б) ни дава $T_\Gamma^\infty(\varnothing) \subseteq \mathrm{diag}(\mathbf{M})$. Но $\varphi \in T_\Gamma^\infty(\varnothing)$, следователно $\varphi \in \mathrm{diag}(\mathbf{M})$, и значи $\mathbf{M}$ е модел на φ. ■

За да докажем теоремата за пълнота на правата изводимост с частни случаи, ще се възползваме от най-малкия модел. Доказателството ще извършим по следния начин. Грубо казано, искаме да видим, че ако псевдоформулата φ е вярна в моделите на Γ, то φ може да се изведе с права изводимост. Но щом φ е вярна в моделите на Γ, то в частност формулата φ ще бъде вярна и в най-малкия модел на Γ. Най-малкият модел има това интересно свойство, че него са верни точно нещата, които могат да се изведат от Γ. Значи щом φ е вярна в най-малкия модел, то φ може да се изведе.

✓ **4.2.26. Теорема за пълнота на правата изводимост с частни случаи.** *Нека* Γ *е логическа програма. Ако някоя псевдоформула се удовлетворява при програма* Γ*, то тя се извежда с права изводимост с частни случаи при програма* Γ*.*

Доказателство. Нека **M** е структурата от теоремата най-малкият модел. Тя е модел на Γ и предикатно обогатяване на **B**. Ако φ е псевдоформула, която се удовлетворява при програма Γ, то φ не използва вграден предикатен символ и **M** е модел на φ. Следователно $\varphi \in \mathrm{diag}(\mathbf{M})$. Но съгласно теоремата за най-малкия модел $\mathrm{diag}(\mathbf{M}) = T_\Gamma^\infty(\varnothing)$ и значи $\varphi \in T_\Gamma^\infty(\varnothing)$, което пък означава, че φ се извежда с права изводимост с частни случаи при програма Γ. ■

4.3. Обратна изводимост

Обратна изводимост с частни случаи

За разлика от правата изводимост, при обратната изводимост използването на частни случаи (засега) няма практически приложения. Въпреки това сега ще разработим теорията на обратната изводимост с частни случаи. Ще направим това, защото е неразумно без подготовка да се гмуркаме изведнъж в дълбокото, тъй като в противен случай може и да не изплуваме. Дори да се интересуваме не от абстрактна теория, от приложенията ѝ, пак има смисъл преди да се впускаме да разработваме теорията на смислената практика, да разработим теория, която макар и да е с безсмислена практика, обаче е по-проста и по-лесна за разбиране.

При правата изводимост използваме клаузите от Γ, които „знаем“, за да доказваме все повече и повече неща — елементите на $T_\Gamma^\infty(\varnothing)$ — и правим това докато успеем да получим желаната цел. При обратната изводимост започваме не с нещата, които знаем, а със запитването, което искаме да докажем, и постепенно го свеждаме до по-лесни за доказване запитвания. Запитванията, които се опитваме да докажем, не са атомарни формули, а конюнкции от атомарни формули, които в дефиниция 4.1.3 нарекохме „запитвания“. При обратната изводимост с частни случаи се използват частни случаи на запитвания, т.е. псевдозапитвания. Да дефинираме какво значи псевдозапитване.

4.3.1. Дефиниция. а) *Псевдозапитване* при сигнатура $\mathfrak{sig}$ е израз от вида

$$\texttt{?-}\ \varphi_1, \varphi_2, \ldots, \varphi_n$$

където $\varphi_1, \varphi_2, \ldots, \varphi_n$ са псевдоформули при сигнатура $\mathfrak{sig}$. Когато $n = 0$ псевдозапитването се нарича *празно псевдозапитване*.

б) Нека структурата $\mathbf{M}$ е обогатяване на $\mathbf{B}$. Едно псевдозапитване е *вярно* в $\mathbf{M}$, ако всичките му псевдоформули са верни в $\mathbf{M}$.

в) Нека Г е логическа програма. Едно псевдозапитване се *удовлетворява* при програма Г, ако всичките му псевдоформули се удовлетворяват при програма Г.

И тъй, при обратната изводимост започваме с някакво псевдозапитване $\texttt{?-}\,\varphi_1, \varphi_2, \ldots, \varphi_n$ и го преработваме, използвайки клаузите от програмата. Да дефинираме как се прави това.

4.3.2. Дефиниция. Нека Г е логическа програма.

а) Казваме че псевдозапитването

$$\texttt{?-}\,\varphi_0, \varphi_1, \varphi_2 \ldots, \varphi_n$$

се свежда едностъпково с обратна изводимост с частни случаи по първи начин към

$$\texttt{?-}\,\psi_1, \psi_2, \ldots, \psi_k, \varphi_1, \varphi_2 \ldots, \varphi_n$$

при програма Г, ако псевдоклаузата $\varphi_0 \texttt{:-}\,\psi_1, \psi_2, \ldots, \psi_k$ е частен случай на клауза от Г (разбира се тук позволяваме $k = 0$).

б) Казваме че псевдозапитването

$$\texttt{?-}\,\varphi_0, \varphi_1, \varphi_2 \ldots, \varphi_n$$

се свежда едностъпково с обратна изводимост с частни случаи по втори начин към

$$\texttt{?-}\,\varphi_1, \varphi_2 \ldots, \varphi_n$$

при програма Г, ако φ_0 е псевдоформула при сигнатурата ВГРАД, която е вярна в структурата $\mathbf{B}$.

4.3.3. Дефиниция. а) Казваме, че псевдозапитването δ *се свежда с обратна изводимост с частни случаи* при програма Г към δ', ако съществува такава редица от псевдозапитвания $\delta_0, \delta_1, \delta_2, \ldots, \delta_m$, че първото псевдозапитване δ_0 е δ, всяко псевдозапитване се свежда едностъпково към следващото и последното псевдозапитване δ_m е δ'.

б) Казваме, че псевдозапитването δ се извежда с обратна изводимост с частни случаи при програма Г, ако δ се свежда към празното псевдозапитване при програма Г.

Да забележим, че всяко псевдозапитване се свежда към себе си. Релацията „свеждане“ представлява рефлексивно и транзитивно затваряне на релацията „едностъпково свеждане“.

✓ **4.3.4. Лема.** *Нека структурата* $\mathbf{M}$ *е модел на логическата програма* Γ *и псевдозапитването* δ *се свежда едностъпково с обратна изводимост при програма* Γ *към* δ'*. Ако* $\mathbf{M} \vDash \delta'$*, то* $\mathbf{M} \vDash \delta$*.*

✓ Доказателство. Да допуснем, че $\mathbf{M} \vDash \delta'$. Трябва да докажем, че $\mathbf{M} \vDash \delta$. Нека псевдозапитването δ е

$$\texttt{?-}\ \varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n$$

Ще разгледаме два случая в зависимост от това по кой начин δ се свежда към δ'.

Ако това става по първи начин (вж. дефиниция 4.3.2), тогава псевдозапитването δ' има вида

$$\texttt{?-}\ \psi_1, \psi_2, \ldots, \psi_k, \varphi_1, \varphi_2, \ldots, \varphi_n$$

и Γ съдържа клауза

$$\varphi_0' \texttt{:-}\ \psi_1', \psi_2', \ldots, \psi_k'$$

с частен случай

$$\varphi_0 \texttt{:-}\ \psi_1, \psi_2, \ldots, \psi_k$$

Нека този частен случай се получава при оценка v. Щом $\mathbf{M} \vDash \delta'$, то в $\mathbf{M}$ са верни псевдоформулите $\psi_1, \psi_2, \ldots, \psi_k, \varphi_1, \varphi_2, \ldots, \varphi_n$. Щом в $\mathbf{M}$ са верни псевдоформулите $\psi_1, \psi_2, \ldots, \psi_k$, то значи формулите $\psi_1', \psi_2', \ldots, \psi_k'$ са верни в $\mathbf{M}$ при оценка v. Щом $\mathbf{M}$ е модел на Γ, то в частност $\mathbf{M}$ е модел и на клаузата $\varphi_0' \texttt{:-}\ \psi_1', \psi_2', \ldots, \psi_k'$, откъдето следва, че в $\mathbf{M}$ формулата φ_0' е вярна при оценка v, от където пък следва, че в $\mathbf{M}$ е вярна псевдоформулата φ_0. Така получаваме, че $\mathbf{M}$ е модел на всяка от псевдоформулите $\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n$, а значи и на псевдозапитването δ.

Ако δ се свежда едностъпково към δ' по втори начин, тогава δ' има вида

$$\texttt{?-}\ \varphi_1, \varphi_2, \ldots, \varphi_n$$

като псевдоформулата φ_0 е с вграден предикатен символ и е вярна в $\mathbf{B}$. Щом φ_0 е вярна в $\mathbf{B}$ и $\mathbf{M}$ е обогатяване на $\mathbf{B}$, то φ_0 е вярна и в $\mathbf{M}$. Псевдоформулите $\varphi_1, \varphi_2, \ldots, \varphi_n$ пък са верни в $\mathbf{M}$, защото са част от псевдозапитването δ', което е вярно в $\mathbf{M}$. Излиза, че $\mathbf{M}$ е модел на всяка от псевдоформулите $\varphi_0, \varphi_1, \varphi_2, \ldots, \varphi_n$, а значи и на псевдозапитването δ.

■

4.3.5. Теорема за коректност на обратната изводимост с частни случаи. *Ако δ се извежда с обратна изводимост с частни случаи при програма Γ, то δ се удовлетворява при програма Γ.*

Доказателство. Нека структурата $\mathbf{M}$ е модел на Γ. Нека редицата, която показва, че δ се свежда с обратна изводимост с частни случаи към празното запитване при програма Γ е $\delta_0, \delta_1, \delta_2, \ldots, \delta_m$. Тъй като в тази редица последното запитване е празното запитване, то $\mathbf{M}$ е негов модел. Тъй като всяко запитване се свежда едностъпково към следващото, от предната лема следва, че $\mathbf{M}$ е модел и на предпоследното запитване, от където получаваме, че $\mathbf{M}$ е модел и на пред-предпоследното и т. н. Значи $\mathbf{M}$ е модел и на първото запитване, т. е. на δ. ∎

Пристъпваме към доказателството на теоремата за пълнота на обратната изводимост с частни случаи. Идеята на доказателството принадлежи на проф. Димитър Скордев.

4.3.6. Дефиниция. Нека Γ е множество от клаузи. Казваме, че псевдоформулата φ е *отстранима* при програма Γ, ако всяко псевдозапитване от вида

$$\texttt{?-}\ \varphi, \psi_1, \psi_2, \ldots, \psi_n.$$

(позволяваме $n = 0$) се свежда с обратна изводимост с частни случаи при програма Γ към псевдозапитването

$$\texttt{?-}\ \psi_1, \psi_2, \ldots, \psi_n.$$

4.3.7. Лема. *Нека Γ е множество от клаузи и $\varphi \in T_\Gamma^\infty(\varnothing)$. Тогава φ е отстранима при програма Γ.*

Доказателство. С индукция по n ще докажем, че лемата е вярна когато $\varphi \in T_\Gamma^n(\varnothing)$.

Когато $n = 0$, няма какво да доказваме, защото $T_\Gamma^0(\varnothing) = \varnothing$.

Да допуснем, че лемата е вярна за n. Нека $\varphi \in T_\Gamma^{n+1}(\varnothing)$, т.е. φ се извежда с частни случаи от $T_\Gamma^n(\varnothing)$. Единият начин това да се случи (вж. дефиниция 4.2.12), е когато φ е псевдоформула при сигнатурата ВГРАД, която е вярна в $\mathbf{B}$. Но тогава отстранимостта на φ следва непосредствено от дефиницията за едностъпково свеждане (дефиниция 4.3.2). Другият начин това да се случи е някоя клауза от Γ да има такъв частен случай

$$\varphi \texttt{:-}\ \chi_1, \chi_2, \chi_3, \chi_4, \ldots, \chi_n \tag{36}$$

че $\chi_1, \chi_2, \ldots, \chi_n \in T^n_\Gamma(\varnothing)$. От индукционното предположение имаме, че псевдоформулите $\chi_1, \chi_2, \ldots, \chi_n$ са отстраними. За да докажем, че φ е отстранима, да изберем произволно псевдозапитване от вида

$$\texttt{?-}\ \varphi, \psi_1, \psi_2, \ldots, \psi_m$$

Използвайки псевдоклаузата (36), виждаме, че това псевдозапитване се свежда едностъпково с обратна изводимост с частни случаи към псевдозапитването

$$\texttt{?-}\ \chi_1, \chi_2, \chi_3, \chi_4, \ldots, \chi_n, \psi_1, \psi_2, \ldots, \psi_m$$

Вече видяхме, че псевдоформулата χ_1 е отстранима, значи горното псевдозапитване се свежда (незадължително едностъпково) към

$$\texttt{?-}\ \chi_2, \chi_3, \chi_4 \ldots, \chi_n, \psi_1, \psi_2, \ldots, \psi_m$$

Псевдоформулата χ_2 също е отстранима и значи това пък псевдозапитване можем да сведем до

$$\texttt{?-}\ \chi_3, \chi_4, \ldots, \chi_n, \psi_1, \psi_2, \ldots, \psi_m$$

По този начин, една по една можем да отстраним всички псевдоформули χ_i и да получим

$$\texttt{?-}\ \psi_1, \psi_2, \ldots, \psi_m.$$

■

4.3.8. Теорема за пълнота на обратната изводимост с частни случаи. *Ако едно псевдозапитване се удовлетворява при програма* Γ*, то то се извежда с обратна изводимост с частни случаи при програма* Γ*.*

Доказателство. Да допуснем, че псевдозапитването $\texttt{?-}\ \varphi_1, \ldots, \varphi_n$ се удовлетворява при програма Γ. От теоремата за пълнота на правата изводимост с частни случаи (теорема 4.2.26) следва, че псевдоформулите $\varphi_1, \varphi_2, \ldots, \varphi_n$ са елементи на $T^\infty_\Gamma(\varnothing)$, и значи тези псевдоформули са отстраними. Щом φ_1 е отстранима, то значи горното псевдозапитване може да се сведе към $\texttt{?-}\ \varphi_2, \varphi_3, \ldots, \varphi_n$. Щом φ_2 е отстранима, то последното псевдозапитване пък може да се сведе към $\texttt{?-}\ \varphi_3, \varphi_4, \ldots, \varphi_n$. Продължавайки по този начин, виждаме, че първоначалното псевдозапитване може да се сведе към празното псевдозапитване, което по дефиниция означава, че то се извежда с обратна изводимост с частни случаи при програма Γ. ■

Логическо програмиране с ограничения

Да видим сега как можем да реализираме обратната изводимост без да се използват частни случаи. Идеята е следната: ще използваме запитвания с ограничения, всяко от които е еквивалентно на безкрайна съвкупност от псевдозапитвания. Например безкрайната съвкупност от псевдозапитванията

$$\texttt{?- p("0", "1")}$$
$$\texttt{?- p("1", "2")}$$
$$\texttt{?- p("2", "3")}$$
$$\texttt{?- p("3", "4")}$$
$$\ldots$$

е еквивалентна на

$$\langle \texttt{?- p(x, y)} \parallel \texttt{y} = \texttt{x} + 1 \rangle$$

Изразът след двойната вертикална черта се нарича „ограничение“.* Ще предполагаме, че имаме алгоритъм за решаване на ограниченията.

Да поясним нещата с точна дефиниция.

✓ **4.3.9. Дефиниция.** а) *Ограничение* ще наричаме формула, която или е равна на $\top$, или представлява конюнкция от атомарни формули от сигнатурата ВГРАД. Ако условно приемем, че $\top$ е конюнкция на нула атомарни формули от сигнатурата ВГРАД, то тогава може да кажем, че ограничение означава конюнкция на нула или повече атомарни формули от ВГРАД. За ограничението $\varphi_1 \,\&\, \varphi_2 \,\&\, \ldots \,\&\, \varphi_n$ ще използваме следния запис:

$$\varphi_1, \varphi_2, \ldots, \varphi_n$$

б) *Състоянието* е формула от вида $\delta \& \varepsilon$, където δ е запитване, а ε — ограничение. За състоянието $\delta \,\&\, \varepsilon$ ще използваме следния запис:

$$\langle \delta \parallel \varepsilon \rangle$$

✓ **4.3.10.** Езиците за логическо програмиране с ограничения работят по следния начин. Да допуснем, че сме задали на компютъра запитване δ. Започваме изпълнението на логическата програма от състояние $\langle \delta \parallel \top \rangle$. След това на всяка стъпка преобразуваме текущото състояние по начина, описан в следващата дефиниция, като идеята на това преобразуване

*На английски „constraint“.

– Обикновено равенството на два безкрайни обекта не може да се установява алгоритмично. Следователно ако искаме в логическата програма да използваме безкрайни типове данни (каквито има напр. в ленивите функционални езици), тогава трябва да работим без равенство.

Преди да дефинираме как работи обратната изводимост с ограничения, ще имаме нужда от следната помощна дефиниция:

4.3.13. Дефиниция. а) Една субституция е *преименуваща*, ако заменя променливи с променливи по биективен начин.

б) Нека φ е клауза. Всички клаузи от вида φs, където s е преименуваща субституция, се наричат *варианти* на клаузата φ.

4.3.14. Пример. Интуитивният смисъл на горната дефиниция се състои в преименуване променливите. Да разгледаме например клаузата

$$\mathtt{p(x,y)\ :\!-\ q(x,x), r(z)}$$

Всяка една от следните клаузи е неин вариант:

$$\mathtt{p(y,x)\ :\!-\ q(y,y), r(z)}$$
$$\mathtt{p(x_1,x_2)\ :\!-\ q(x_1,x_1), r(x_3)}$$
$$\mathtt{p(x_1,y')\ :\!-\ q(x_1,x_1), r(z_1)}$$

Клаузата

$$\mathtt{p(x_1,y')\ :\!-\ q(x',x'), r(z_1)}$$

не е вариант, защото променливата $\mathtt{x}$ е преименувана някъде като $\mathtt{x}_1$ а другаде като $\mathtt{x}'$. Клаузата

$$\mathtt{p(x,x)\ :\!-\ q(x,x), r(x)}$$

също не е вариант, защото е получена посредством субституция v, за която $v(\mathtt{x}) = v(\mathtt{y}) = v(\mathtt{z}) = \mathtt{x}$, а такава субституция не е биективна функция.

Вече може да пристъпим към дефиницията на обратната изводимост с ограничения.

✓ **4.3.15. Дефиниция.** Нека Г е логическа програма и ни е дадено състояние

$$\langle \mathtt{?\text{-}\ p}(\tau_1,\ldots,\tau_n), \varphi_1, \varphi_2, \ldots, \varphi_k \parallel \psi_1, \ldots, \psi_l \rangle$$

Тогава:

а) Ако клаузата

$$\mathtt{p}(\sigma_1, \ldots, \sigma_n) \mathtt{:-} \chi_1, \chi_2, \ldots, \chi_m$$

е вариант на някоя клауза от логическата програма Γ, то даденото състояние *се свежда едностъпково с обратна изводимост* по първи начин при програма Γ до

$$\langle \mathtt{?-}\ \chi_1, \chi_2, \ldots, \chi_m, \varphi_1, \varphi_2, \ldots, \varphi_k \parallel \tau_1 = \sigma_1\ \ldots, \tau_n = \sigma_n, \psi_1, \ldots, \psi_l \rangle$$

б) Ако p е символ от ВГРАД, то даденото състояние *се свежда едностъпково с обратна изводимост* по втори начин при програма Γ до

$$\langle \mathtt{?-}\ \varphi_1, \varphi_2, \ldots, \varphi_k \parallel \mathtt{p}(\tau_1, \ldots, \tau_n), \psi_1, \ldots, \psi_l \rangle$$

С други думи, ако първата атомарна формула от текущото запитване е с вграден предикатен символ, то просто я прехвърляме в ограничението.

4.3.16. ***Забележка:*** а) Когато предикатният символ p не е от ВГРАД, в общия случай текущото състояние се преобразува по недетерминиран начин. Това е така, защото в програмата може да има много клаузи от вида

$$\mathtt{p}(\sigma_1, \ldots, \sigma_n) \mathtt{:-} \chi_1, \chi_2, \ldots, \chi_m$$

По принцип това означава, че при многопроцесорни или многоядрени компютри имаме възможност да изпълняваме паралелно тези алтернативни свеждания на текущото състояние. За съжаление езикът пролог не се възползва от паралелизма на съвременните компютри — при него клаузите се обработват последователно, според реда, в който са написани в програмата.

б) Причината, поради използваме не самите клаузи от Γ, а техни варианти, е следната — променливите, които се срещат в текущото състояние, нямат по смисъл нищо общо с променливите, които са били използвани, когато пишем програмата. Затова, за да не възникнат проблеми, работим не непосредствено с клаузите от програмата, а с варианти на клаузите, в които променливите са преименувани, така че да се различават от променливите в състоянието.

в) Да забележим, че дефиницията е формулирана по такъв начин, че когато ограничението в дадено състояние е неизпълнимо, то тогава всички състояния, към които можем да го сведем, също имат

неизпълнимо ограничение. Макар в горната дефиниция това да не е изрично споменато, на практика може да игнорираме състоянията, за чиито ограничения решаващият алгоритъм ни каже, че са неизпълними.

Следващата дефиниция формализира това, което в 4.3.10 изказахме неформално.

✓ **4.3.17. Дефиниция.** а) Казваме, че състоянието $\langle \delta \parallel \varepsilon \rangle$ се свежда към $\langle \delta' \parallel \varepsilon' \rangle$ при програма Г, ако съществува редица от състояния

$$\langle \delta_0 \parallel \varepsilon_0 \rangle, \langle \delta_1 \parallel \varepsilon_1 \rangle, \ldots, \langle \delta_n \parallel \varepsilon_n \rangle$$

в която първото състояние е $\langle \delta \parallel \varepsilon \rangle$, последното е $\langle \delta' \parallel \varepsilon' \rangle$ и всяко състояние се свежда едностъпково към следващото.

б) Нека е дадена логическа програма Г. Казваме, че при дадената програма *запитването* $\texttt{?-}\,\varphi_1, \ldots, \varphi_n$ *се извежда с обратен извод с отговор* v, ако v е частична оценка, дефинирана само за променливите, които се срещат в запитването, състоянието

$$\langle \texttt{?-}\,\varphi_1, \ldots, \varphi_n \parallel \top \rangle$$

се свежда при програма Г към

$$\langle \texttt{?-} \parallel \varepsilon \rangle$$

и ограничението ε от последното състояние е вярно в $\mathbf{B}$ при някаква оценка v', която е разширение на v.

Да илюстрираме с два примера работата на език за логическо програмиране с ограничения.

4.3.18. Пример. Нека универсумът на структурата $\mathbf{B}$ е множество, чиито елементи са хората Адам, Каин и Авел. Нека имаме три вградени символа за константи `адам`, `авел` и `каин`, които се интерпретират в $\mathbf{B}$ по очаквания начин. Нека освен това има и два вградени предикатни символа $=$ и $\neq$, които също се интерпретират в $\mathbf{B}$ по очаквания начин, т.е. съответно като равенство и неравенство. Да разгледаме логическа програма, състояща се от следните три клаузи:

$\texttt{син}(\texttt{авел}, \texttt{адам})$
$\texttt{син}(\texttt{каин}, \texttt{адам})$
$\texttt{брат}(X, Y)\ \texttt{:-}\ X \neq Y, \texttt{син}(X, Z), \texttt{син}(Y, Z)$

✓ **4.3.21. Дефиниция.** Нека φ е формула и Γ е множество от клаузи. Ще казваме че формулата φ *се удовлетворява* с отговор v при програма Γ, ако v е частична оценка в $\mathbf{B}$, дефинирана само за променливите от φ и φ е вярна при частичната оценка v във всички структури за $\mathfrak{sig}$, които са едновременно предикатно обогатяване на $\mathbf{B}$ и модел на Γ.

Теоремите за коректност и пълнота на обратната изводимост ще покажат, че едно запитване се извежда с обратен извод с отговор v тогава и само тогав, това запитване се удовлетворява с отговор v. Доказателството на тези теореми ще направим като ги сведем към теоремите за коректност и пълнота за обратната изводимост с частни случаи. Връзката между удовлетворяването на запитване φ с отговор v и удовлетворяването на неговия частен случай при оценка v е ясна:

4.3.22. Твърдение. *Нека δ е запитване, Γ — логическа програма и v е частична оценка в $\mathbf{B}$, дефинирана само за променливите, срещащи се в δ. Запитването δ се удовлетворява при програма Γ с отговор v тогава и само тогава, когато частният случай на δ при оценка v се удовлетворява при програма Γ.*

Доказателство. По същество доказателството представлява просто упражнение върху дефинициите. Случаят когато $\delta = \top$ е очевиден. Нека $\delta = \psi_1 \,\&\, \psi_2 \,\&\, \ldots \,\&\, \psi_n$. Нека частните случаи на $\delta, \psi_1, \psi_2, \ldots, \psi_n$ при оценка v са съответно $\delta', \psi_1', \psi_2', \ldots, \psi_n'$. Тъй като $\delta' = \psi_1' \,\&\, \psi_2' \,\&\, \ldots \,\&\, \psi_n'$, то δ' ще се удовлетворява при програма Γ тогава и само тогава, когато всяка от псевдоформулите ψ_i' се удовлетворява при програма Γ. Това е така тогава и само тогава, когато всяка една от тези псевдоформули е вярна във всяка структура, която е едновременно обогатяване на $\mathbf{B}$ и модел на Γ. Последното е така тогава и само тогава, когато всяка от атомарните формули $\psi_1, \psi_2, \ldots, \psi_n$ е вярна при оценка v във всяка структура, която е едновременно обогатяване на $\mathbf{B}$ и модел на Γ (твърдение 4.2.9). Това пък е така когато запитването δ е вярно при оценка v във всяка структура, която е едновременно обогатяване на $\mathbf{B}$ и модел на Γ. Тъй като v е дефинирана само за променливите, срещащи се в δ, то последното се случва тогава и само тогава, когато запитването δ се удовлетворява при програма Γ с отговор v. ∎

По-сложно се доказва връзката между извеждането на дадено състояние с отговор v и извеждането на частния случай на това състояние при частична оценка v. Най-напред ще установим връзка между едностъпковото свеждане на състояния и едностъпковото свеждане на частни случаи. Едва след това от тук като следствие ще получим нужното.

в която всяко състояние се свежда едностъпково към следващото и псевдозапитването ζ_n е частен случай при частичната оценка v на състоянието $\langle \delta_n \parallel \varepsilon_n \rangle$. Следователно съществува оценка v', която е обогатяване на v, при която този частен случай се получава. Ограничението ε_n, разбира се, трябва да бъде вярно при тази оценка. Остава само да забележим, че от $\zeta_n = \top$ следва $\delta_n = \top$ и значи δ_0 се извежда с обратна изводимост с отговор v. ∎

4.3.27. Теорема за коректност на обратната изводимост с ограничения. *Ако запитването δ се извежда с обратен извод при програма Γ с отговор v, то δ се удовлетворява при програма Γ с отговор v.*

Доказателство. Току-що доказаното следствие показва, че δ има частен случай ζ при частичната оценка v, който се извежда с обратна изводимост с частни случаи. От теоремата за коректност на обратната изводимост с частни случаи следва, че ζ се удовлетворява при програма Γ. От твърдение 4.3.22 следва, че δ се удовлетворява при програма Γ с отговор v. ∎

4.3.28. Теорема за пълнота на обратната изводимост с ограничения. *Ако запитването δ се удовлетворява при програма Γ с отговор v, то δ се извежда с обратен извод при програма Γ с отговор v.*

Доказателство. От твърдение 4.3.22 следва, че δ има частен случай ζ при частичната оценка v, който се удовлетворява при програма Γ. От теоремата за пълнота на обратната изводимост с частни случаи следва, че ζ се извежда с обратна изводимост. От по-горе доказаното следствие следва, че δ се извежда с обратна изводимост с отговор v. ∎

4.4. Ербранови структури

В предните раздели се уговорихме, че разполагаме с фиксирана структура **B**. Тази структура определя какви вградени функции и предикати има в езика за логическо програмиране, а от нейния универсум разбираме с какви обекти може да работим в програмата. За логическата програма може да кажем, че надгражда над структурата **B**, т.е. приемаме, че имаме наготово всичко, предоставено от **B** и използваме логика, за да дефинираме нови предикати. Затова почти всички дефиниции в предните раздели споменаваха структурата **B**. Така например за едно запитване φ казахме, че се удовлетворява при дадена

✓ **4.4.3. Дефиниция.** Структурата $\mathbf{H}$ е *ербранова*, ако:

- Универсумът на $\mathbf{H}$ е множеството от всички термове, които не съдържат променливи.
- За всеки символ за константа $\mathtt{c}$, стойността му е самата константа, т.е. $\mathtt{c}^{\mathbf{H}} = \mathtt{c}$. (Да забележим, че $\mathtt{c}$ е терм без променливи.)
- За всеки n-местен функционален символ $\mathtt{f}$ и елементи $\tau_1, \tau_2, \ldots, \tau_n$ на универсума на $\mathbf{H}$ имаме

$$\mathtt{f}^{\mathbf{H}}(\tau_1, \tau_2, \ldots, \tau_n) = \mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n) \tag{53}$$

(Да забележим, че щом $\tau_1, \tau_2, \ldots, \tau_n$ са от универсума на $\mathbf{H}$, то те са термове без променливи, а значи и $\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$ е терм без променливи, т.е. елемент на универсума.)

4.4.4. Дефиниция. Множеството от всички термове без променливи ще наричаме *ербранов универсум*.

4.4.5. Обърнете внимание, че скобите от двете страни на равенство (53) имат напълно различен смисъл. Скобите отляво ги използваме, за да покажем, че даваме $\tau_1, \tau_2, \ldots, \tau_n$ като аргументи на функцията $\mathtt{f}^{\mathbf{H}}$. Скобите отдясно пък са просто символи — вторият и последният символ на терма $\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$. Също така напълно различен е и смисълът на запетаите. Използваме запетаите отляво, за да разделим аргументите на функцията $\mathtt{f}^{\mathbf{H}}$. За разлика от тях, запетаите отдясно са просто символи, които се срещат някъде в терма $\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$.

***Задача* 48:** Нека сигнатурата съдържа поне един символ за константа. Докажете подробно, че съществуват ербранови структури.

Тъй като елементите на универсума на една ербранова структура са термове без променливи, то всяка оценка в тази структура съпоставя на променливите термове без променливи. Това означава, че всяка оценка в ербранова структура представлява субституция и значи можем да я прилагаме по два различни начина:

- можем да пресметнем стойността на един терм при тази оценка;
- можем да си мислим, че оценката е субституция и да я приложим към този терм.

Следващото твърдение показва, че и в двата случая ще получим един и същ резултат, т.е. $[\![\tau]\!]^{\mathbf{H}} v = \tau v$.

✓ **4.4.6. Твърдение.** *Нека* $\mathbf{H}$ *е ербранова структура и* v *е оценка в* $\mathbf{H}$. *Тогава стойността на кой да е терм* τ *в* $\mathbf{H}$ *при оценка* v *е равна на резултата от прилагането на субституцията* v *към терма* τ.

✓ Доказателство. Ще докажем твърдението с индукция по терма τ. Да припомним, че с $[\![\tau]\!]^{\mathbf{H}}v$ означаваме стоността на τ при оценка v в структурата $\mathbf{H}$, а τv е резултатът от прилагането на субституцията v към τ.

Ако $\tau = \mathtt{x}$ е променлива, то стойността на τ при оценка v е стойността на $\mathtt{x}$ при оценка v, т.е. $v(\mathtt{x})$. Да приложим субституцията v към $\mathtt{x}$ означава да заместим $\mathtt{x}$ с $v(\mathtt{x})$ и значи отново получаваме $v(\mathtt{x})$.

Ако $\tau = \mathtt{c}$ е символ за константа, то по дефиниция 3.2.2 б) стойността на τ в $\mathbf{H}$ при коя да е оценка е $\mathtt{c}^{\mathbf{H}} = \mathtt{c}$. Резултатът от прилагането на коя да е субституция към терма $\mathtt{c}$ също е $\mathtt{c}$.

Нека $\tau = \mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)$. Стойността на τ в $\mathbf{H}$ при оценка v е

$$[\![\tau]\!]^{\mathbf{H}}v = \mathtt{f}^{\mathbf{H}}([\![\tau_2]\!]^{\mathbf{H}}v, [\![\tau_2]\!]^{\mathbf{H}}v, \ldots, [\![\tau_n]\!]^{\mathbf{H}}v)$$

Понеже структурата е ербранова, последното е равно на

$$\mathtt{f}([\![\tau_1]\!]^{\mathbf{H}}v, [\![\tau_2]\!]^{\mathbf{H}}v, \ldots, [\![\tau_n]\!]^{\mathbf{H}}v)$$

което съгласно индукционното предположение е равно на

$$\mathtt{f}(\tau_1 v, \tau_2 v, \ldots, \tau_n v) = (\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n))v = \tau v$$

■

✓ **4.4.7. Следствие.** *Ако* $\mathbf{H}$ *е ербранова структура и* τ *е терм без променливи, то стойността на* τ *в* $\mathbf{H}$ *при коя да е оценка е* τ*.*

✓ Доказателство. Нека v е произволна оценка в $\mathbf{H}$. Съгласно твърдение 4.4.6 стойността на τ в $\mathbf{H}$ при оценка v е равна на резултата от прилагането на субституцията v към τ. Но τ не съдържа променливи, които субституциите могат да заместват и значи като приложим v към τ получаваме пак τ. ■

✓ ***Задача* 49:** Нека $\mathbf{H}$ е ербранова структура и оценката v в $\mathbf{H}$ е такава, че $v(\mathtt{x}) = \mathtt{f}(\mathtt{f}(\mathtt{c}))$ и $v(\mathtt{y}) = \mathtt{g}(\mathtt{c}, \mathtt{f}(\mathtt{a}))$. Кои от скобите и кои от запетаите в следния израз са символи и кои не са символи:

$$\mathtt{f}(\mathtt{g}^{\mathbf{H}}(v(\mathtt{x}), (\mathtt{g}(\mathtt{a}, \mathtt{y}))v))$$

Колко леви скоби и колко запетаи се съдържат в стойността на този израз?

За безкванторните формули вместо твърдение 4.4.6 можем да докажем следното твърдение:

✓ **4.4.8. Твърдение.** *Нека* $\mathbf{H}$ *е ербранова структура,* v *е оценка в* $\mathbf{H}$ *и* φ *е безквантворна формула. Тогава формулата* φ *е вярна в* $\mathbf{H}$ *при оценка* v *тогава и само тогава, когато формулата* φv *е вярна в* $\mathbf{H}$.

Доказателство. С индукция по формулата φ.

Нека най-напред φ е атомарна и $\varphi = \mathtt{p}(\tau_1, \tau_2, \ldots, \tau_n)$. Тъй като структурата е ербранова, съгласно твърдение 4.4.6 стойността на τ_i в $\mathbf{H}$ при оценка v е $\tau_i v$. Термовете $\tau_i v$ не съдържат променливи и значи съгласно твърдение 4.4.7 тяхната стойност също е $\tau_i v$. Следователно както атомарната формула $\varphi = \mathtt{p}(\tau_1, \tau_2, \ldots, \tau_n)$, така и атомарната формула $\varphi v = \mathtt{p}(\tau_1 v, \tau_2 v, \ldots, \tau_n v)$ е вярна в $\mathbf{H}$ при оценка v тогава и само тогава, когато е истина $\mathtt{p}^{\mathbf{H}}(\tau_1 v, \tau_2 v, \ldots, \tau_n v)$. Остава само да отбележим, че съгласно твърдение 3.4.9, да бъде формулата φv вярна при оценка v е същото, каквото тя да бъде вярна при коя да е оценка.

Нека $\varphi = \bot$. Тогава $\mathbf{H} \vDash \varphi[v]$ е невярно и $\mathbf{H} \vDash \varphi v$ също е невярно, защото $\varphi v = \bot$.

Нека $\varphi = \psi_1 \,\&\, \psi_2$. Тогава

$$\begin{aligned}
\mathbf{H} \vDash \varphi[v] &\longleftrightarrow \mathbf{H} \vDash \psi_1 \,\&\, \psi_2[v] \\
&\longleftrightarrow \mathbf{H} \vDash \psi_1[v] \text{ и } \mathbf{H} \vDash \psi_2[v] \\
&\longleftrightarrow \mathbf{H} \vDash (\psi_1 v) \text{ и } \mathbf{H} \vDash (\psi_2 v) \qquad \text{(инд. предп.)} \\
&\longleftrightarrow \mathbf{H} \vDash (\psi_1 v) \,\&\, (\psi_2 v) \longleftrightarrow \mathbf{H} \vDash ((\psi_1 \,\&\, \psi_2) v) \\
&\longleftrightarrow \mathbf{H} \vDash (\varphi v)
\end{aligned}$$

Случаите когато $\varphi = \psi_1 \vee \psi_2$ и $\varphi = \psi_1 \Rightarrow \psi_2$ са аналогични. ∎

4.4.9. Лема. *Нека* $\mathbf{M}$ *е произволна структура. Тогава съществува ербранова структура* $\mathbf{H}$ *и силен хомоморфизъм* $h \colon \mathbf{H} \to \mathbf{M}$.

Доказателство. Дефиниция 4.4.3 за ербранова структура ни казва какъв е универсумът на $\mathbf{H}$ — множеството от всички термове, които не съдържат променливи. Пак от дефиницията разбираме и как в $\mathbf{H}$ се интерпретират символите за константи и функционалните символи. Остава единствено да определим как в $\mathbf{H}$ се интерпретират предикатните символи. Да отложим засега това.

Хомоморфизмът h трябва да изобразява всеки терм без променливи в елемент на универсума на $\mathbf{M}$. Да дефинираме $h(\tau)$ да бъде стойността на τ в $\mathbf{M}$ (тъй като τ не съдържа променливи, не е нужно да уточняваме коя е оценката, вж. твърдение 3.2.5). Да докажем, че това наистина е силен хомоморфизъм.

Първо, по дефиниция h изобразява елементите на универсума на $\mathbf{H}$ е елементи на универсума на $\mathbf{M}$.

Второ, за всеки символ за константа $\mathtt{c}$:

$$h(\mathtt{c}^{\mathbf{H}}) = h(\mathtt{c}) = \mathtt{c}$$

Освен това, за всеки n-местен функционален символ $\mathtt{f}$ (в долните равенства с $[\![\ldots]\!]^{\mathbf{M}}$ е означена стойността в $\mathbf{M}$ на терм без променливи):

$$\begin{aligned} h(\mathtt{f}^{\mathbf{H}}(\tau_1, \tau_2, \ldots, \tau_n)) &= h(\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)) \\ &= [\![\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n)]\!]^{\mathbf{M}} \\ &= \mathtt{f}^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}, [\![\tau_2]\!]^{\mathbf{M}}, \ldots, [\![\tau_n]\!]^{\mathbf{M}}) \\ &= \mathtt{f}^{\mathbf{M}}(h(\tau_1), h(\tau_2), \ldots, h(\tau_n)) \end{aligned}$$

И накрая, за всеки n-местен предикатен символ $\mathtt{p}$ може просто да дефинираме

$$\mathtt{p}^{\mathbf{H}}(\tau_1, \tau_2, \ldots, \tau_n) \overset{\text{def}}{\longleftrightarrow} \mathtt{p}^{\mathbf{M}}([\![\tau_1]\!]^{\mathbf{M}}, [\![\tau_2]\!]^{\mathbf{M}}, \ldots, [\![\tau_n]\!]^{\mathbf{M}})$$

защото тогава

$$\mathtt{p}^{\mathbf{H}}(\tau_1, \tau_2, \ldots, \tau_n) \longleftrightarrow \mathtt{p}^{\mathbf{M}}(h(\tau_1), h(\tau_2), \ldots, h(\tau_n))$$

■

✓ **4.4.10. Теорема („малка“ теорема на Ербран).** *а) Нека* Γ *е множество от безкванторни формули и* φ *е безкванторна формула. Тогава* φ *е изпълнима във всеки модел на* Γ *тогава и само тогава, когато* φ *е изпълнима във всеки ербранов модел на* Γ.

б) Множество Γ *от безкванторни формули има модел тогава и само тогава, когато то има ербранов модел.*

Доказателство. **(а)** Ясно е, че ако φ е изпълнима във всеки модел на Γ, то в частност φ ще бъде изпълнима и във всеки ербранов модел на Γ. За да докажем обратната посока, да допуснем, че φ е изпълнима във всеки ербранов модел на Γ и да изберем произволен (не задължително ербранов) модел на Γ. Трябва да докажем, че φ е изпълнима в $\mathbf{M}$.

От лема 4.4.9 намираме ербранова структура $\mathbf{H}$ и хомоморфизъм $h\colon \mathbf{H} \to \mathbf{M}$. От твърдение 3.5.9 а) следва, че $\mathbf{H}$ е модел на Γ. Тъй като φ е изпълнима във всеки ербранов модел на Γ, то значи φ е изпълнима в $\mathbf{H}$, следователно от твърдение 3.5.9 б) получаваме, че φ е изпълнима в $\mathbf{M}$.

(б) Може да сведем към (а). Нека $\varphi = \bot$. Тъй като φ е неизпълнима формула, то φ е изпълнима във всеки модел на Г тогава и само тогава, когато Г няма модели и φ е изпълнима във всеки ербранов модел на Г тогава и само тогава, когато Г няма ербранови модели. ■

4.4.11. Може да използваме „малката“ теорема на Ербран, за да сведем логическото програмиране без структура $\mathbf{B}$ към логическото програмиране със структура $\mathbf{B}$. Нека единственият предикатен символ в сигнатурата ВГРАД е равенството. И нека $\mathbf{B}$ е ербрановата структура за тази сигнатура, в която символът за равенство се интерпретира като равенство (т.е. универсумът на $\mathbf{B}$ съдържа термовете без променливи, а символите за константи и функционалните символи се интерпретират съгласно дефиниция 4.4.3). При така дефинирана структура $\mathbf{B}$ е вярно следното следствие:

✓ **4.4.12. Следствие.** *Нека* Г *е логическа програма, която не съдържа символа за равенство и* φ *е запитване, което също не съдържа символа за равенство. Тогава* φ *се удовлетворява от* Г *(вж. дефиниция 4.4.1) тогава и само тогава, когато съществува отговор, с който при тази програма запитването* φ *се удовлетворява над* $\mathbf{B}$*. (вж. дефиниция 4.3.21).*

Доказателство. φ се удовлетворява от Г тогава и само тогава, когато φ е изпълнима във всеки модел на Г. Съгласно малката теорема на Ербран това се случва тогава и само тогава, когато φ е изпълнима във всеки ербранов модел на Г. Тъй като Г и φ не съдържат символа за равенство, то интерпретацията на този символ е без значение и значи последното се случва тогава и само тогава, когато φ е изпълнима във всеки ербранов модел на Г, в който символа за равенство се интерпретира като равенство. Но една структура е ербранова и символа за равенство в нея се интерпретира като равенство тогава и само тогава, когато тя е обогатяване на $\mathbf{B}$. Следователно горното се случва тогава и само тогава, когато φ е изпълнима във всеки модел на Г, който е обогатяване на $\mathbf{B}$. Последното е равносилно с това да съществува отговор, с който при програма Г запитването φ се удовлетворява. ■

4.4.13. Дефиниция. Ако φ е безквантorна формула и s е субституция, заместваща всяка променлива с терм без променливи, то формулата φs се нарича *затворен частен случай* на формулата φ при субституция s.

4.4.14. Теорема („средна“ теорема на Ербран). *Множество* Г *от безквантorни формули е изпълнимо тогава и само тогава, когато*

предикати, то ще получим език за логическо програмиране с ограничения, използващ ербранова структура $\mathbf{B}$.*

Ще опишем алгоритъм, който може да се използва за решаване на ограничения, когато структурата $\mathbf{B}$ е дефинирана както в 4.4.11. От съображения за ефективност повечето версии на пролог използват леко видоизменен вариант на този алгоритъм, който не винаги работи коректно при ербранова структура $\mathbf{B}$, но не извършва т. н. occurs check и затова е по-ефективен. Вж. края на този раздел.

Да припомним, че съгласно дефиниция 4.3.9 а) ограниченията представляват конюнкции от атомарни формули от сигнатурата ВГРАД. В нашия случай единственият предикатен символ в сигнатурата ВГРАД е $=$, което означава, че ограниченията представляват конюнкции от равенства между термове:

$$\tau_1 = \sigma_1 \,\&\, \tau_2 = \sigma_2 \,\&\, \ldots \,\&\, \tau_n = \sigma_n$$

Затова може да си мислим, че ограниченията представляват системи от нула (когато ограничението е $\top$) или повече уравнения, които решаваме в структурата $\mathbf{B}$.

✓ **4.5.1. Дефиниция.** Казваме, че оценката v е *решение на ограничението* φ, ако $\mathbf{B} \vDash \varphi[v]$.

Разбира се, v е решение на дадено ограничение тогава и само тогава, когато всички уравнения в него са верни при оценка v.

4.5.2. Дефиниция. За две ограничения казваме, че са *еквивалентни* в $\mathbf{B}$, ако имат едни и същи решения.

✓ **4.5.3. Дефиниция.** Казваме, че едно ограничение е *решено* относно променливата $\mathtt{x}$, ако тази променлива се среща точно веднъж в ограничението и то в уравнение от вида $\mathtt{x} = \tau$. (От тук в частност следва, че $\mathtt{x}$ не се среща в τ, нито в което и да е друго уравнение от ограничението). За уравнението $\mathtt{x} = \tau$ казваме, че е *решаващо*.

Алгоритъмът за унификация, към чиято дефиниция пристъпваме сега, представлява метод, посредством който за всяко ограничение може да намерим еквивалентно на него ограничение, в което всички уравнения са решаващи.

*Дори и да изхвърлим „нелогическите“ предикати, в пролог пак ще останат много вградени предикати освен равенството. Това обаче от теоретична гледна точка е без значение, защото има начин тези предикати да се дефинират с логическа програма и значи не е нужно да считаме, че те са вградени.

4.5.4. Дефиниция. Следните преобразувания на ограничения ще наричаме *решаващи преобразувания*:

Първо решаващо преобразуване. Ако в ограничението има уравнение от вида

$$\tau = \mathtt{x}$$

където $\mathtt{x}$ е променлива, а термът τ не е променлива, то заменяме това уравнение с

$$\mathtt{x} = \tau$$

Второ решаващо преобразуване. Ако в ограничението има уравнение от вида

$$\mathtt{x} = \mathtt{x}$$

където $\mathtt{x}$ е променлива, то отстраняваме това уравнение от ограничението.

Трето решаващо преобразуване. Ако в ограничението има уравнение от вида

$$\mathtt{f}(\tau_1, \tau_2, \ldots, \tau_n) = \mathtt{f}(\sigma_1, \sigma_2, \ldots, \sigma_n)$$

то заменяме това уравнение с

$$\tau_1 = \sigma_1 \,\&\, \tau_2 = \sigma_2 \,\&\, \ldots \,\&\, \tau_n = \sigma_n$$

Ако в ограничението има уравнение от вида

$$\mathtt{c} = \mathtt{c}$$

където $\mathtt{c}$ е символ за константа, то отстраняваме това уравнение от ограничението.

Четвърто решаващо преобразуване. Ако в ограничението има уравнение от вида

$$\mathtt{x} = \tau$$

което не е решаващо и променливата $\mathtt{x}$ не се среща в терма τ, то заместваме навсякъде в останалите уравнения променливата $\mathtt{x}$ с τ.

4.5.5. Алгоритъмът за унификация се състои в следното: да прилагаме решаващи преобразувания към ограничението по произволен начин докато може. Следващото твърдение показва, че ако към някое ограничение не може повече да прилагаме решаващи преобразувания, то в него със сигурност или всички уравнения ще бъдат решаващи, или ограничението ще съдържа уравнение, което няма да има решения, и значи цялото ограничение няма да има решения.

(твърдение 3.3.13), за произволен терм σ

$$[\![\sigma]\!]^{\mathbf{H}} w = [\![\sigma[\mathtt{x} \coloneqq \tau]]\!]^{\mathbf{H}} v$$

където оценката w е дефинирана с равенството

$$w(\mathtt{z}) = [\![\mathtt{z}[\mathtt{x} \coloneqq \tau]]\!]^{\mathbf{H}} v$$

Но $v(\mathtt{x}) = [\![\tau]\!]^{\mathbf{H}} v$, а когато $\mathtt{z} \neq \mathtt{x}$, то $\mathtt{z}[\mathtt{x} \coloneqq \tau] = \mathtt{z}$ и значи оценките v и w съвпадат. Следователно за произволен терм σ

$$[\![\sigma]\!]^{\mathbf{H}} v = [\![\sigma[\mathtt{x} \coloneqq \tau]]\!]^{\mathbf{H}} v$$

т.е. стойността на кой да е терм при оценка v си остава същата, ако към терма приложим субституцията $\mathtt{x} \coloneqq \tau$. Тъй като четвъртото решаващо преобразуване представлява прилагане на точно тази субституция, то значи v или е решение на ограничението както преди, така и след преобразуването, или v не е решение на ограничението нито преди, нито след преобразуването. ∎

✓ **4.5.9.** В логическото програмиране алгоритъмът за унификация се използва по следния начин. След като го приложим към ограничението на текущото състояние, то или получаваме ограничение, за което знаем, че няма решение, или получаваме ограничение, в което всяко уравнение е решаващо. В първия случай цялото състояние няма решение и повече не го използваме. Във втория случай нека сме получили състояние от вида

$$\langle \texttt{?-}\ \varphi_1, \varphi_2, \ldots, \varphi_k \parallel \mathtt{x}_1 = \tau_1, \mathtt{x}_2 = \tau_2, \ldots, \mathtt{x}_n = \tau_n \rangle$$

Да приложим към всяка от формулите $\varphi_1, \varphi_2, \ldots, \varphi_k$ субституцията

$$\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau_1, \tau_2, \ldots, \tau_n$$

Ако означим така получените формули с $\varphi'_1, \varphi'_2, \ldots, \varphi'_k$, то ще получим състоянието

$$\langle \texttt{?-}\ \varphi'_1, \varphi'_2, \ldots, \varphi'_k \parallel \mathtt{x}_1 = \tau_1, \mathtt{x}_2 = \tau_2, \ldots, \mathtt{x}_n = \tau_n \rangle$$

в което никоя от променливите $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ не се среща в запитването $\texttt{?-}\ \varphi'_1, \varphi'_2 \ldots, \varphi'_k$. Това означава, че изпълнението на програмата може да продължи без да се интересуваме повече от ограничението. Чак когато пролог намери отговор на първоначално поставеното запитване от потребителя, може да използваме равенствата в ограничението, за да изведем отговор. И тъй като може да прилагаме алгоритъма за унификация винаги, когато решим, това означава, че изчисленията може да се извършват, използвайки състояния, от чиито ограничения не се интересуваме преди да завърши работата на програмата.

Тук няма да доказваме, че замяната на формулите $\varphi_1, \varphi_2, \ldots, \varphi_k$ с $\varphi'_1, \varphi'_2, \ldots, \varphi'_k$ по описания по-горе начин не променя по същество начина, по който работи една логическа програма, спрямо това, което вече бе описано в раздел 4.3. Ще се задоволим да докажем семантичната коректност на тази замяна. Тази коректност следва от следното твърдение:

4.5.10. Твърдение. *Нека променливите* $\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n$ *не се срещат в термовете* $\tau_1, \tau_2, \ldots, \tau_n$. *Ако структурата* $\mathbf{M}$ *е обогатяване на* $\mathbf{B}$, *то формулата*

$$\varphi \,\&\, \mathtt{x}_1 = \tau_1 \,\&\, \mathtt{x}_2 = \tau_2 \,\&\, \ldots \,\&\, \mathtt{x}_n = \tau_n$$

е вярна в $\mathbf{M}$ *при някоя оценка* v *тогава и само тогава, когато при същата оценка е вярна формулата*

$$\varphi[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau_1, \tau_2, \ldots, \tau_n] \,\&\, \mathtt{x}_1 = \tau_1 \,\&\, \mathtt{x}_2 = \tau_2 \,\&\, \ldots \,\&\, \mathtt{x}_n = \tau_n$$

Доказателство. Доказателството е аналогично на разглеждането на решаващите преобразувания от четвърти тип в твърдение 4.5.8, само че прилагаме лемата за субституциите за формули, а не за термове.

Ако някоя от двете формули в условието на твърдението е вярна при оценка v, то $v(\mathtt{x}_i) = [\![\tau_i]\!]^{\mathbf{H}} v$. Съгласно лемата за субституциите (твърдение 3.3.17), за формулата φ е вярно

$$\mathbf{M} \vDash \varphi[w] \longleftrightarrow \mathbf{M} \vDash \varphi[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau_1, \tau_2, \ldots, \tau_n][v]$$

където оценката w е дефинирана с равенството

$$w(\mathtt{z}) = [\![\mathtt{z}[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau_1, \tau_2, \ldots, \tau_n]]\!]^{\mathbf{H}} v$$

Но $v(\mathtt{x}_i) = [\![\tau_i]\!]^{\mathbf{H}} v$, а когато $\mathtt{z} \notin \{\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n\}$, то

$$\mathtt{z}[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau_1, \tau_2, \ldots, \tau_n] = \mathtt{z}$$

и значи оценките v и w съвпадат. Следователно за формулата φ е имаме

$$\mathbf{M} \vDash \varphi[v] \longleftrightarrow \mathbf{M} \vDash \varphi[\mathtt{x}_1, \mathtt{x}_2, \ldots, \mathtt{x}_n \coloneqq \tau_1, \tau_2, \ldots, \tau_n][v]$$

От тук следва исканото. ∎

Примери

Да разгледаме с конкретен пример как работи пролог.

✓ **4.5.11. Пример.** Нека програмата се състои от следните клаузи:

$$\mathtt{p(a)}\ \mathtt{:-}\ \mathtt{p(c)}. \tag{54}$$
$$\mathtt{p}(X)\ \mathtt{:-}\ \mathtt{r}(X). \tag{55}$$
$$\mathtt{r(a)}. \tag{56}$$

Ще приложим тази програма към запитването

$$\mathtt{?-}\ \mathtt{p(a)}$$

Клауза (54) свежда първоначалното състояние

$$\langle \mathtt{?-}\ \mathtt{p(a)} \parallel \top \rangle \tag{57}$$

към състоянието

$$\langle \mathtt{?-}\ \mathtt{p(c)} \parallel \mathtt{a} = \mathtt{a} \rangle$$

Като приложим третото решаващо преобразувание към $\mathtt{a} = \mathtt{a}$, това състояние се опростява до

$$\langle \mathtt{?-}\ \mathtt{p(c)} \parallel \top \rangle$$

Единствената клауза, която не свежда това състояние към състояние с неизпълнимо ограничение, е клауза (55). Посредством нея то се свежда до

$$\langle \mathtt{?-}\ \mathtt{r}(X_1) \parallel \mathtt{c} = X_1 \rangle$$

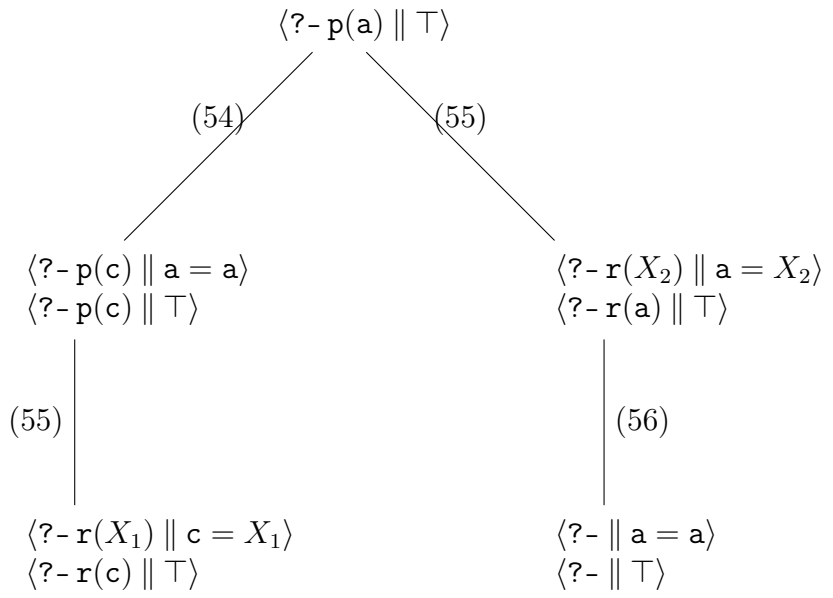
Алгоритъмът за унификация свежда това ограничение към $X_1 = \mathtt{c}$. След прилагане на субституцията $X_1 \coloneqq \mathtt{c}$ към запитването, променливата X_1 става излишна, така че състоянието се опростява до

$$\langle \mathtt{?-}\ \mathtt{r(c)} \parallel \top \rangle$$

Никоя клауза не може да се използва за това запитване, така че то остава неудовлетворено.

За запитването (57) имаме още една възможност — да приложим клауза (55). Посредством нея то се свежда до

$$\langle \mathtt{?-}\ \mathtt{r}(X_2) \parallel \mathtt{a} = X_2 \rangle$$



Фиг. 14.

Алгоритъмът за унификация свежда това ограничение към $X_2 = \mathtt{a}$. След прилагане на субституцията $X_1 := \mathtt{a}$ към запитването, променливата X_2 става излишна, така че състоянието се опростява до

$$\langle \mathtt{?-\ r(a)} \parallel \top \rangle$$

Клауза (56) свежда това състояние до

$$\langle \mathtt{?-} \parallel \mathtt{a} = \mathtt{a} \rangle$$

Третото решаващо преобразуване елиминира равенството $\mathtt{a} = \mathtt{a}$, така че получаваме успешното състояние

$$\langle \mathtt{?-} \parallel \top \rangle$$

Всички свеждания от пример 4.5.11 могат да бъдат изобразени с дърво, както това е направено във фигура 14. Във върховете на това дърво сме записали по две състояния. Първото се получава преди да приложим алгоритъмът за унификация, а второто — след това. Пълнотата означава, че ако първоначалната заявка се удовлетворява, то това дърво със сигурност ще съдържа успешен клон (т.е. клон, чиито състояния представляват извод).

Интерпретаторът на пролог обхожда това дърво в дълбочина, започвайки от най-левите клонове. В случая пролог ще обходи най-напред

левия клон на дървото, където няма да намери решение на задачата, и след това ще се прехвърли на десния клон.

Когато в дървото има безкраен ляв клон, пролог ще се зацикли без да открие успешен клон. Да разгледаме още един пример.

✓ **4.5.12. Пример.** Нека програмата този път се състои от следните клаузи:

$$\mathtt{p}(X)\,\mathtt{:-}\,\mathtt{p}(\mathtt{c}). \tag{58}$$
$$\mathtt{p}(X)\,\mathtt{:-}\,\mathtt{r}(X). \tag{59}$$
$$\mathtt{r}(\mathtt{a}). \tag{60}$$

Дървото на извод, което се получава от тази програма при запитване `?- p(a)`, е изобразено на фигура 15. За по-добра прегледност, в това дърво са изобразени само запитванията в състоянията, които се получават след прилагане на алгоритъма за унификация. Вижда се, че това дърво съдържа безкраен ляв клон. Пролог ще тръгне по този клон и никога няма да открие десния и успешен клон.

Това показва, че при пролог е от значение редът, в който са записани клаузите. Когато дървото на извод е крайно, тогава при лошо подреждане на клаузите пролог може да работи значително по-бавно. Например при дървото на фигура 14 пролог ненужно хаби време, за да обхожда левия и неуспешен клон. Много по-лошо е положението, когато дървото е безкрайно. В този случай пролог може до безкрайност да се движи по някой безкраен клон, както се случи при дървото от фигура 15.

В първата от тези програми проблемът се оправя, ако се разменят клаузи (54) и (55), а във втората — като се разменят клаузи (58) и (59). За съжаление обаче, в някои случаи пролог се зацикля без значение в какъв ред подреждаме клаузите.

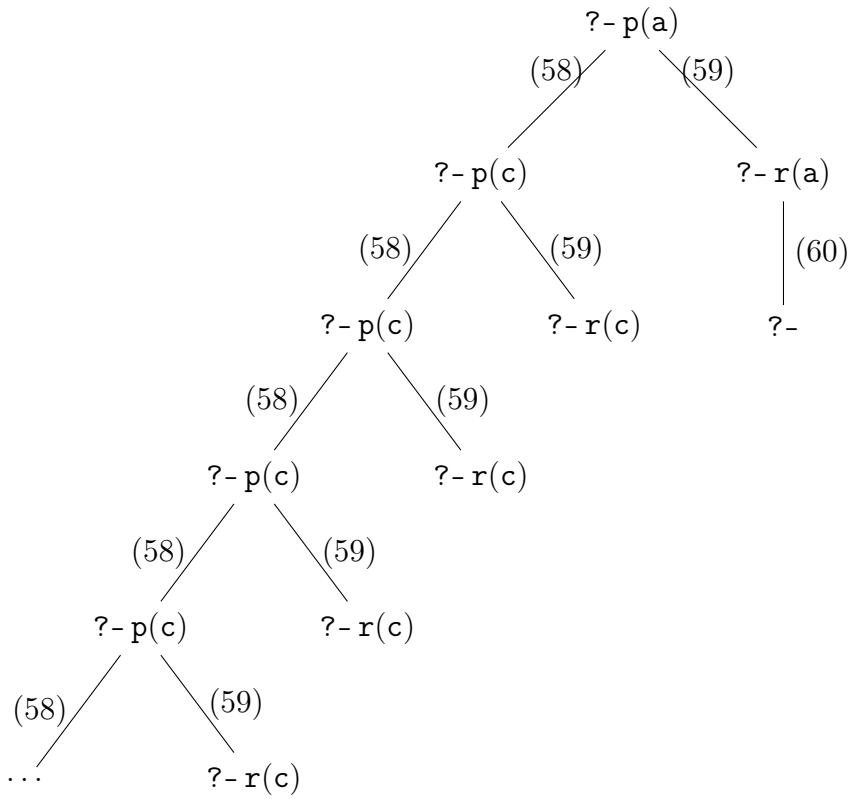
Да разгледаме още един пример.*

✓ **4.5.13. Пример.** Нека програмата се състои от следните клаузи:

$$\mathtt{p}(\mathtt{a},\mathtt{c}). \tag{61}$$
$$\mathtt{p}(X,Y)\,\mathtt{:-}\,\mathtt{p}(Y,X). \tag{62}$$
$$\mathtt{p}(X,Y)\,\mathtt{:-}\,\mathtt{p}(X,Z),\,\mathtt{p}(Z,Y). \tag{63}$$

По принцип запитването `?- p(c, c)` може да се удовлетвори с обратен извод. Едно успешно свеждане на това запитване до празното запитване

*Автор на примера е проф. Димитър Скордев [26].



Фиг. 15.

е дадено на фигура 16. Въпреки това, както и да подреждаме клаузите в тази програма, винаги ще се оказва, че отляво на този успешен клон, а и изобщо отляво на кой да е успешен клон, в дървото ще има безкраен клон. Затова при тази програма пролог ще се зацикля без значение как подреждаме клаузите в програмата.

Унификация без occurs check

Ако разгледаме решаващите преобразувания (вж. дефиниция 4.5.4) от алгоритъма за унификация, може да забележим, че първото, второто и третото решаващо преобразование могат да се изпълнят на компютър за константно време (т.е. време, което не зависи от размера на термовете). При четвъртото решаващо преобразование имаме уравнение от вида $\mathtt{x} = \tau$ и трябва да проверим дали променливата $\mathtt{x}$ се среща в терма τ. Тази проверка, наречена на английски occurs check, не може да се извърши за константно време, защото зависи от големината на терма τ.

запитвания сме в състояние да изразим прости формули като елементарните дизюнкции, от тук би следвало, че и въпроси, формулирани посредством произволни формули, могат да бъдат сведени към клаузи и запитвания.

За съжаление, оказва се, че това не е възможно. С помощта на клаузи и запитвания могат да се представят не произволни, а т.н. хорнови елементарни дизюнкции.

Да видим защо това е така.

4.6.2. Дефиниция. Една елементарна дизюнкция е *хорнова*, ако в нея има най-много един литерал без отрицание.

✓ **4.6.3. Твърдение.** *а) Нека* Γ *е множество от клаузи и* ζ *е запитване. Ако използваме класическата логика, то запитването* ζ *се удовлетворява при програма* Γ *тогава и само тогава, когато множеството* $\Gamma \cup \{\neg\zeta\}$ *е неизпълнимо.*

б) Може да се намери такова множество от хорнови елементарни дизюнкции,[*] *че при използване на некласическата логика то е неизпълнимо тогава и само тогава, когато е неизпълнимо множеството* $\Gamma \cup \{\neg\zeta\}$.

✓ Доказателство. **(а)** Нека Γ е множеството от клаузи и ζ е запитване, за което се питаме дали се удовлетворява при програма Γ. По дефиниция да се удовлетворява ζ означава за всеки модел на Γ да има оценка v, при която запитването ζ е вярно (дефиниция 4.4.1). Ако разсъждаваме неконструктивно, това е така тогава и само тогава, когато не съществува модел на Γ, при който ζ е лъжа при всяка оценка. С други думи, тогава и само тогава, когато множеството $\Gamma \cup \{\neg\zeta\}$ е неизпълнимо.

(б) Доказахме първата част от твърдението. За да докажем втората част, да разгледаме най-напред тривиалния случай когато ζ е празното запитване. По дефиниция 4.1.3, $\zeta = \top$, значи $\neg\zeta = \neg\top$, следователно $\neg\zeta$ е винаги невярна формула и значи множеството $\Gamma \cup \{\neg\zeta\}$ е неизпълнимо все едно каква е програмата Γ. Значи ще ни свърши работа кое да е неизпълнимо множество от хорнови елементарни дизюнкции.[**]

[*] Смисълът тук е това, че това множество от елементарни дизюнкции може да бъде намерено конструктивно, т.е. посредством алгоритъм.

[**] Например ако в сигнатурата има едноместен предикатен символ $\mathtt{p}$ и символ за константа $\mathtt{c}$, то множеството $\{\mathtt{p}(\mathtt{c}), \neg\mathtt{p}(\mathtt{c})\}$ е неизпълнимо множество от две елементарни дизюнкции.

Сега да разгледаме случая когато ζ е непразно запитване. Да видим как ще изглеждат елементите на множеството $\Gamma \cup \{\neg\zeta\}$ когато ги изразим посредством елементарни дизюнкции.

Елементите на Γ са клаузи. Всяка клауза от вида

$$\psi \texttt{ :- } \varphi_1, \varphi_2, \ldots, \varphi_n$$

при $n \geqq 1$ представлява различен запис на формулата

$$\varphi_1 \,\&\, \varphi_2 \,\&\, \ldots \,\&\, \varphi_n \Rightarrow \psi$$

Ако преобразуваме тази формула в конюнктивна нормална форма по начина, описан в доказателствата на твърдения 3.7.8 и 3.7.30, ще получим формула, състояща се от една единствена елементарна дизюнкция:

$$\neg\varphi_1 \vee \neg\varphi_2 \vee \cdots \vee \neg\varphi_n \vee \psi$$

Когато пък $n = 0$, то клаузата представлява просто формулата ψ и значи пак е елементарна дизюнкция от горния вид (само че $n = 0$). Специфичното при тази елементарна дизюнкция е това, че тя съдържа произволен брой (може и нула) литерали с отрицание, но точно един литерал без отрицание. И така, направихме следното важно наблюдение: всяка клауза може да бъде представена посредством хорнова елементарна дизюнкция.

Да видим сега как можем да представим $\neg\zeta$ посредством елементарни дизюнкции. Нека ζ е произволно запитване (където $n \geqq 1$)

$$\texttt{?- } \varphi_1, \varphi_2, \ldots, \varphi_n$$

По дефиниция това запитване е различен запис на формулата

$$\varphi_1 \,\&\, \varphi_2 \,\&\, \ldots \,\&\, \varphi_n$$

Като сложим отпред отрицание получаваме

$$\neg(\varphi_1 \,\&\, \varphi_2 \,\&\, \ldots \,\&\, \varphi_n)$$

След като преобразуваме тази формула в конюнктивна нормална форма по начина, описан в доказателствата на твърдения 3.7.8 и 3.7.30, получаваме елементарната дизюнкция

$$\neg\varphi_1 \vee \neg\varphi_2 \vee \cdots \vee \neg\varphi_n$$

Специфичното при тази елементарна дизюнкция е това, че тя съдържа произволен брой литерали с отрицание и не съдържа нито един литерал без отрицание. Следователно тази елементарна дизюнкция също е хорнова. ■

И така, направихме следното важно наблюдение: въпросът дали дадено запитване се удовлетворява при някакво множество от клаузи е еквивалентен на въпроса дали множество от хорнови елементарни дизюнкции е неизпълнимо.

В този раздел ще опишем методът на резолюциите, посредством който компютрите могат да „разсъждават“ използвайки произволни елементарни дизюнкции, а не само хорнови. Но за да ни бъде по-удобно, вместо с елементарни дизюнкции, ще работим с по-просто устроени обекти, наречени дизюнкти.*

4.6.4. Дефиниция. а) *Дизюнкт* е крайно (може и празно) множество от литерали.

б) Един дизюнкт е верен в структура $\mathbf{M}$ при оценка v, ако не е невярно, че в дизюнкта можем да намерим верен в $\mathbf{M}$ при оценка v литерал.

в) Един дизюнкт е (тъждествено) верен в структура $\mathbf{M}$, ако е верен при всяка оценка в $\mathbf{M}$.

г) Множество от дизюнкти е *изпълнимо*, ако съществува структура, в която са тъждествено верни всички дизюнкти от множеството. Множество от дизюнкти е *неизпълнимо*, ако не е изпълнимо.

4.6.5. *Забележка*: Току що дадената дефиниция показва, че дизюнкт от вида

$$\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$$

е еквивалентен на формулата

$$\neg\neg(\lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_n)$$

Разбира се, когато разсъждаваме, използвайки класическата логика, тази формула е еквивалентна на елементарната дизюнкция

$$\lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_n$$

Тъй като дизюнктът $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$ по отношение на класическата логика е еквивалентен на елементарната дизюнкция $\lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_n$, то значи от твърдение 3.7.31 моментално получаваме следното следствие:

*Терминът „дизюнкт“ се използва само в България. В другите страни вместо него се използва терминът „клауза“ (clause). Разбира се, тъй като нещата, които тук наричаме „клаузи“, в другите страни също се наричат клаузи, то от контекста трябва да познаваме за какъв вид клаузи става въпрос.

4.6.6. Следствие. *Ако в сигнатурата разполагаме с достатъчно неизползвани символи за константи и функционални символи, то за всяко крайно множество*[*] *от формули можем да намерим такова крайно множество от дизюнкти, че (от гледна точка на класическата логика) първоначалното множество е изпълнимо тогава и само тогава, когато е изпълнимо множеството от дизюнкти.*

По подобен начин може да установим, че много от нещата, които сме доказали за формули, са верни и когато използваме дизюнкти. Например от „малката“ теорема на Ербран за безквантворни формули може да получим като следствие „малка“ теорема на Ербран за дизюнкти:

4.6.7. Теорема („малка“ теорема на Ербран за дизюнкти). *Множество* Γ *от дизюнкти има модел тогава и само тогава, когато има ербранов модел.*

Доказателство. Празният дизюнкт е еквивалентен на формулата $\bot$. Освен това, всеки дизюнкт

$$\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$$

е еквивалентен на формулата[**]

$$\neg\neg(\lambda_1 \vee \lambda_2 \vee \cdots \vee \lambda_n)$$

Горните две наблюдения показват, че можем лесно да изведем тази теорема като следствие от „малката“ теорема на Ербран за безквантворни формули 4.4.10. Нека Γ' е множеството от формулите, съответстващи на дизюнктите от Γ. Множеството Γ е изпълнимо тогава и само тогава, когато е изпълнимо Γ' защото дизюнктите от Γ са еквивалентни на съответните формули от Γ'. Съгласно „малката“ теорема на Ербран за безквантворни формули, множеството Γ' има модел тогава и само тогава, когато то има ербранов модел. Следователно също и множеството Γ има модел тогава и само тогава, когато то има ербранов модел. ∎

Следващото твърдение е упражнение как да се прави отрицание.

✓ **4.6.8. Твърдение.** *Един дизюнкт е тъждествено верен в структура* $\mathbf{M}$ *тогава и само тогава, когато не съществува оценка в* $\mathbf{M}$*, при която всички литерали от дизюнкта са неверни.*

[*] Твърдението е вярно и за безкрайни множества, но тук няма да доказваме това.

[**] Разбира се, когато използваме класическата логика можем да премахнем двойното отрицание.

✓ Доказателство. Да допуснем, че дизюнктът δ е тъждествено верен в $\mathbf{M}$. Трябва да докажем, че не съществува оценка в $\mathbf{M}$, при която всички литерали от дизюнкта са неверни. Ами да допуснем, че такава оценка съществува, т.е. оценка, при която всички литерали са неверни. В такъв случай при тази оценка няма да бъде вярно, че дизюнктът съдържа верен литерал, а това противоречи на допускането, че дизюнктът е тъждествено верен в $\mathbf{M}$.

Обратната посока също е бърза. Да допуснем, че не съществува оценка в $\mathbf{M}$, при която всички литерали от дизюнкта са неверни. Но това означава, че каквато и оценка да вземем, няма да е вярно, че при тази оценка всички литерали са неверни. Щом не е вярно, че всички литерали са неверни, значи не е невярно, че можем да намерим верен литерал. Което пък означава, че дизюнктът е верен при коя да е оценка в $\mathbf{M}$. ■

4.6.9. Дефиниция. Празното множество от литерали се нарича *празен дизюнкт*. За да бъде по-ясно, че работим не с какво да е празно множество, а с дизюнкт, вместо с $\varnothing$, означаваме празния дизюнкт с някой от символите $\square$, ■ или $\{\}$.

Следващото твърдение показва, че празният дизюнкт има същия смисъл, както винаги невярната формула $\bot$.

4.6.10. Твърдение. *Празният дизюнкт не е тъждествено верен в никоя структура.*

Доказателство. Ами да си изберем структура $\mathbf{M}$ и оценка v. Вярно ли е твърдението, че в дизюнкта можем да намерим литерал, който е верен при така избраните оценка и структура? Ами очевидно не, защото в дизюнкта не можем да намерим какъвто и да е литерал. Щом като това твърдение е невярно, значи не е вярно, че то не е невярно. ■

Резолюция с частни случаи

Когато разработвахме теорията на изводимостта с клаузи бе по-удобно най-напред да дефинираме правата и обратната изводимост с частни слуаи и едва след това да видим как можем да обобщим метода, така че той да работи с произволни клаузи. При метода на резолюциите положението е аналогично – по-лесно е най-напред да докажем теореми за коректност и пълнота при работа с дизюнкти, които са частни случаи и не съдържат променливи, и едва след това от тук да получим като следствие теореми за коректност и пълнота в общия случай.

при субституцията $[\mathtt{x}, \mathtt{y} \coloneqq \mathtt{f(c)}, \mathtt{a}]$ е дизюнктът

$$\{\mathtt{p(f(c), f(c))}, \neg\mathtt{q(f(a), f(c))}\}$$

✓ ***Задача* 50:** Вярно ли е че всеки частен случай на дизюнкт ε съдържа точно толкова литерали, колкото и ε?

Съгласно „малката“ теорема на Ербран 4.6.7 едно множество от дизюнкти има модел тогава и само тогава, когато то има ербранов модел. Следващото твърдение пък показва, че когато работим в ербранова структура всеки дизюнкт е еквивалентен на множеството от своите частни случаи.

4.6.13. Твърдение. *Дизюнктът ε е верен в ербрановата структура* **H** *при оценка v тогава и само тогава, когато εv (т.е. частният случай на ε при оценка v) е верен в* **H**.

Доказателство. Дизюнктът ε е верен в **H** при оценка v тогава и само тогава, когато не е вярно, че в ε няма литерал, който е верен в **H** при оценка v.

Частният случай εv е верен в **H** при някаква оценка w тогава и само тогава, когато не е вярно, че в εv няма литерал, който е верен в **H** при оценка w.

Литералите на εv са точно литералите от вида λv, където $\lambda \in \varepsilon$. Съгласно твърдение 4.4.8 литералът λ е верен в **H** при оценка v тогава и само тогава, когато λv е верен в **H** (все едно при каква оценка, защото няма променливи). ∎

Идеята на метода на резолюциите е следната. Най-напред използваме следствие 4.6.6, за да сведем първоначалния въпрос, от който се интересуваме, към въпроса дали някакво множество от дизюнкти е неизпълнимо. След това започваме да генерираме всевъзможни резолвенти. Ако след краен брой стъпки се стигне до празния дизюнкт, то значи множеството от дизюнкти е неизпълнимо. В противен случай то е изпълнимо.

Тъй като в този подраздел ще работим с частни случаи, т.е. с дизюнкти без променливи, ще отложим дефиницията на резолвента. Вместо това ще дефинираме по-простото понятие „непосредствена резолвента“, което се оказва достатъчно когато дизюнктите са без променливи.

Твърдение 4.6.13 показва, че поне на теория непосредствената резолюция може да се използва и с произволни дизюнкти по следния начин. Ако Γ е произволно множество от дизюнкти, нека Γ' е множеството от всички частни случаи на дизюнкти от Γ. Тогава ако искаме да проверим

дали Γ е изпълнимо, може да приложим непосредствената резолюция към Γ'. Но макар на теория това да върши работа, на практика така се получава безполезен метод, защото дори в сигнатурата да има само един единствен функционален символ, пак ербрановият универсум ще бъде безкраен и значи множеството Γ' е безкрайно.

4.6.14. Дефиниция. а) Нека са дадени дизюнктите

$$\{\varphi\} \cup \delta' \text{ и } \{\neg\varphi\} \cup \delta''$$

За дизюнкта $\delta' \cup \delta''$ ще казваме, че е тяхна *непосредствена резолвента*.

б) Когато $\varphi \notin \delta'$ и $\neg\varphi \notin \delta''$ непосредствената резолвента е *стриктна*.

Интуитивно, резолвентата се прави като „задраскаме“ една двойка противоположни литерали от двата дизюнкта и обединим останалите.

✓ **4.6.15. Пример.** Нека

$$\delta_1 = \{\mathtt{p(c)}, \mathtt{q(c)}, \mathtt{r(a)}\}$$

и

$$\delta_2 = \{\neg\mathtt{p(c)}, \neg\mathtt{q(c)}, \mathtt{r(b)}\}$$

Ако „задраскаме“ двойката литерали $\mathtt{p(c)}$ и $\neg\mathtt{p(c)}$ получаваме непосредствената резолвента

$$\{\mathtt{q(c)}, \mathtt{r(a)}, \neg\mathtt{q(c)}, \mathtt{r(b)}\}$$

Ако пък „задраскаме“ двойката $\mathtt{q(c)}$ и $\neg\mathtt{q(c)}$ получаваме непосредствената резолвента

$$\{\mathtt{p(c)}, \mathtt{r(a)}, \neg\mathtt{p(c)}, \mathtt{r(b)}\}$$

Важно е да се разбере, че дефиницията не ни позволява да „задраскаме“ едновременно и двете двойки $\mathtt{p(c)}$ и $\neg\mathtt{p(c)}$ и $\mathtt{q(c)}$ и $\neg\mathtt{q(c)}$, така че дизюнктът $\{\mathtt{r(a)}, \mathtt{r(b)}\}$ не е резолвента.

Горните две резолвенти са не само непосредствени, но и стриктни. Резолвентите, които не са стриктни, се получават по по-неестествен начин. Да забележим, че

$$\delta_1 = \{\mathtt{p(c)}\} \cup \{\mathtt{p(c)}, \mathtt{q(c)}, \mathtt{r(a)}\}$$

и

$$\delta_2 = \{\neg\mathtt{p(c)}\} \cup \{\neg\mathtt{p(c)}, \neg\mathtt{q(c)}, \mathtt{r(b)}\}$$

Това означава, че дизюнктът

$$\{\mathtt{p(c)}, \mathtt{q(c)}, \mathtt{r(a)}, \neg\mathtt{p(c)}, \neg\mathtt{q(c)}, \mathtt{r(b)}\}$$

е непосредствена резолвента на δ_1 и δ_2. Тази резолвента не е стриктна.