SBORNÍK PRACÍ FILOZOFICKÉ FAKULTY BRNĚNSKÉ UNIVERZITY
STUDIA MINORA FACULTATIS PHILOSOPHICAE UNIVERSITATIS BRUNENSIS
B 26/27, 1979–1980

JAN ŠTĚPÁN

CHARAKTERISTIKA PŘÍKAZU IF POMOCÍ MODALIT

Programovací jazyky jako prostředky sloužící k bezprostřednímu využívání výpočetních systémů lze chápat jako primitivní slovníky, jejichž prvky (při dodržení syntaktických pravidel) lze popisovat formou symbolických programů libovolné algoritmy vhodné pro strojové zpracování. Algoritmus jako výpočetní návod (postup) reprezentuje (složenou) normu, ale protože jde o posloupnost instrukcí, jejichž provedení vede k dosažení vytčeného cíle, jsou tyto jeho složky také normami. V jazykovém vyjádření pak jde o normativní věty. V symbolických programech jsou normativními větami přímo příkazy[1] příslušného programového jazyka.[2] Zde se budeme zabývat jedním z takových příkazů a pokusíme se o jeho formálně logickou formulaci.

Příkaz IF (podmíněný příkaz) vyskytující se v různých syntaktických podobách ve všech vyšších programovacích jazycích zde budeme chápat (bez újmy na obecnosti) jako „jednostrannou alternativu", tj. schematicky „IF ‚podmínka' THEN ‚příkaz'". Na tento tvar lze převést všechny ostatní varianty tohoto příkazu obecně i v konkrétních aplikacích.

Jakkoliv podmíněný příkaz svým vnějším vzhledem — jazykovým vyjádřením připomíná implikaci („jestliže ‚něco' pak ‚něco jiného'"), nemůže jít o logickou spojku (už proto, že je to příkaz). Přesnější charakteristikou by bylo, že jde o normotvorný (nebo příkazotvorný) funktor. Nabízí se zde srovnání s podmínkovou normativní větou[3], v níž by ‚podmínka' odpovídala hypotéze a ‚příkaz' dospozici. Zde však podobnost ostatně skutečně jen vnější končí, ať už by šlo o případnou sankci, tak i pokud jde o vnitřní strukturu podmíněného příkazu a mechanismus jeho provádění.

---

[1] V úvahu připadají pouze příkazy imperativní — příkazy ve vlastním smyslu — na rozdíl od pomocných příkazů deklaračních, které pouze zajišťují správnou činnost těch předchozích.

[2] Jde vlastně spíše o normativní funkce (formy). Tyto výrazy totiž obsahují (vedle konstant) obecná jména k nimž je přiřazena třída možných hodnot — tedy volné proměnné. V průběhu výpočtu — při provádění programu — však tato jména vždy reprezentují nějakou konkrétní hodnotu ze svého oboru, takže můžeme mluvit o normativních větách.

[3] Viz např. O. Weinberger: *Logika*, Praha 1964, str. 147 a n.

Označíme-li — realizaci podmínky C
— realizaci příkazu S
— realizaci sankce T

(u těchto „realizací" lze hovořit o pravdivostních hodnotách), pak v notaci zmíněné práce by se podmíněný příkaz zapsal jako

$$C > !S$$

a tento výraz by označoval primární normu. Příslušná sankční norma pak vypadá takto:

$$C \wedge \neg S > !T \qquad (1)$$

Avšak u podmíněného příkazu vůbec nemůže nastat případ, který je popsán výrazem $C \wedge \neg S$, tedy k realizaci sankce by nikdy nedošlo.[4]

Formulace příkazu IF v termínech logiky normativních vět je tedy neadekvátní. Dojde-li ke splnění podmínky C, nemůže nedojít k realizaci příkazu S (dochází k ní nutně). Lze tedy chápat splnění podmínky C jako příčinu realizace příkazu S (následek) — jde vlastně o příčinný vztah.

Vyjdeme-li ze skutečnosti, že nemůže být splněn „antecedent" výrazu (1) a současně nesplněn jako „konsekvent", nabízí se možnost formulovat podmíněný příkaz jako striktní implikaci, neboť její definice přesně odpovídá této vlastnosti příkazu IF ($p \supset q = df \neg M (p \wedge \neg q)$), tj.

$$C \supset S \qquad (2)$$

jako zkratku formálního zápisu vlastnosti

$$\neg M (C \wedge \neg S).$$

Dokonce lze tvrdit, že splnění podmínky C vede nutně k realizaci příkazu S

$$C \supset \neg M \neg S.$$

Tím však ještě nejsou postiženy všechny vlastnosti podmíněného příkazu. Jestliže není splněna podmínka C, není realizován příkaz S (striktně)

$$\neg C \supset \neg S, \qquad (3)$$

protože to prostě není možné

$$\neg C \supset \neg MS.$$

Z (2) a (3) užitím transpozice plyne striktní ekvivalence

$$C = S$$

která je sice intuitivně přijatelná, ovšem jen do jisté míry, jelikož nevystihuje onu „propustnost" obsaženou v podmínce; to, že obě realizace (už nastalé) jsou vzájemně podmíněny, ale realizace podmínky je jaksi silnější, představuje příčinu provedení resp. neprovedení celého příkazu IF.

---

[4] Provedení podmíněného příkazu totiž probíhá tak, že je otestována podmínka a v případě, že je splněna, je proveden i příslušný příkaz; v případě, že splněna není, příkaz není proveden a podmíněný příkaz jako celek je neúčinný.

Potíž je zřejmě v tom, že aplikace transpozice na (3) není patrně zcela adekvátní. V transponované striktní implikaci z (3)

$$S \supset C$$

je zaměněna příčina s následkem.

Že užití transpozice není možné vyplývá už ze syntaktické definice podmíněného příkazu. Podmínka totiž je v praxi reprezentována zpravidla booleovským výrazem, který může nabývat pravdivostních hodnot, což v případě libovolného příkazu možné není.[5]

Má-li tedy formálně logická formulace podmíněného příkazu přesně vystihovat jeho podstatu i funkci, musí být respektovány jeho charakteristické vlastnosti, které byly předchozími formulacemi opomenuty. Tyto požadavky splňují následující definice:

D1 $$p \multimap q = \text{df } p \wedge \neg M((p \wedge \neg q) \vee (\neg p \wedge q)),$$

ovšem pouze pro případ, kdy ‚příkaz' je nepodmíněný. Je-li reprezentován podmíněným příkazem resp. sekvencí takových příkazů, je třeba definici upravit do obecnějšího tvaru:

D2 $$p_1 \multimap (p_2 \multimap \ldots \multimap (p_n \multimap q)\ldots) = \text{df } (p_1 \wedge p_2 \wedge \ldots \wedge p_n) \wedge \neg M((p_1 \wedge p_2 \wedge \ldots p_n \wedge \neg q) \vee (\neg(p_1 \wedge p_2 \wedge \ldots p_n) \wedge q)))$$

Z této definice přímo vyplývají vlastnosti, které jsou obdobou vlastností klasické implikace:

V1 $$(p \multimap (p \multimap q)) \leftrightarrow (p \multimap q)$$

V2 $$(p \multimap (q \multimap r)) \leftrightarrow ((p \wedge q) \multimap r)$$

V3 $$(p \multimap (q \multimap r)) \leftrightarrow (q \multimap (p \multimap r))$$

Další vlastnosti zřejmé z definice již „klasickou" obdobu nemají, jsou pouze jakousi doplňující charakteristikou podmíněného příkazu:

V4 $$(p \multimap q) \vee (\neg p \multimap \neg q)$$

V5 $$(p \multimap q) \leftrightarrow \neg(\neg p \multimap \neg q)$$

Je ovšem ještě třeba omezit důsledky definice tak, abychom nedostali neadekvátní tvrzení. Jde o respektování skutečnosti, že ‚podmínku' nelze ani ztotožňovat, ani zaměňovat s příkazem

A1 $$\llcorner M(p \multimap p)$$

A2 $$(p \multimap q) \supset \neg M(q \multimap p)$$

A3 $$p \supset (p \multimap q)$$

---

[5] Proto zde uvažujeme místo příkazu jeho realizaci, u níž jsme pak oprávněni mluvit o pravdivostní hodnotě. Vlastně ani takový booleovský výraz nereprezentuje přímo pravdivostní hodnotu — je totiž obecně obdobou výrokové funkce — pravdivostní hodnoty nabude teprve po udělení hodnot „volným" proměnným (viz pozn. 2). Proto i zde hovoříme o realizaci (podmínky).

Tyto dvě zásady musí být respektovány při případné aplikaci pravidla substituce (oborem podmíněného příkazu je kartézský součin množiny všech možných podmínek s množinou všech možných příkazů). Pravidlo odloučení však platí bez omezení:

PO
$$\frac{A \multimap B \quad A}{B}$$

dokonce v silnější verzi — z předpokladů lze soudit na nutnou platnost „konsekventu":

PO′
$$\frac{A \multimap B \quad A}{NB}$$

Obě tato pravidla (s ohledem na V5) platí i pro případ, že obě proměnné v první premise jsou negovány, tj.:

$$\frac{\neg A \multimap \neg B \quad A}{B} \qquad \frac{\neg A \multimap \neg B \quad A}{NB}$$

resp. je negována druhá premisa a závěr, tj.:

$$\frac{A \multimap B \quad \neg A}{\neg B} \qquad \frac{A \multimap B \quad \neg A}{N\neg B \text{ resp. } \neg MB}$$

Vzhledem k podané definici budou platné také následující úsudky:

$$\frac{A \multimap B \quad A}{N(A \wedge B)} \qquad \frac{A \multimap B \quad \neg A}{\neg M(A \wedge B)}$$

Neobvyklost některých z předchozích úsudků je dána tím, že spojka $\multimap$ má postihnout normativní charakter podmíněného příkazu. Podmíněný příkaz je v tomto směru ojedinělý v rámci takových normativních soustav jakými jsou symbolické programy, pokud k jejich studiu přistupujeme jako v této práci. Jestliže totiž uvažujeme u libovolného jiného (imperativního) příkazu jeho realizaci, musí být nutně vždy pravdivá (jako výrok), jelikož každý nepodmíněný příkaz je vždy proveden. Je tedy možné pouze logickou reprezentací podmíněného příkazu obohatit výrazové prostředky modální logiky.

THE CHARACTERISTICS OF THE IF-STATEMENT BY MEANS OF THE MODALITIES

The paper shows one of the possible ways of how formal logic can formulate imperative statements in programming languages, especially the IF-statement. Owing to its form, the IF-statement (in the most general form: IF „condition" THEN „statement") remings us of implication (which is based on the fact that any unconditioned imperative statement can be considered in terms of its realization, which can be ascribed a truth-value; so far this comparison is justified), but the IF-statement cannot be adequately expressed by means of any known type of implication.

The proposed definition takes fully into account that the „condition" is a condition both necessary and sufficient for the realization of whole IF-statement, and — moreover — the fulfilment of the „condition" and the non-realization of the „statement" or vice versa cannot arise at the same time. Apart from this, it is necessary to postulate further constraints. This specific „implication" cannot follow the law of identity, and the „antecedent" and the „consequent" cannot be interchanged (i. e., it is impossible for the „condition" and the „statement" to merge). The paper also discusses the properties of the IF-statement and formulates the derivation rules, hence showing in which the above connective is either similar to, or different from, the classical implication.