

Macroprocesoare pentru asigurarea portabilității

STAGE-2

În aceeași perioadă am intrat în contact cu o altă piesă importantă a „puzzle-ului portabilității”: STAGE-2, macroprocesorul universal dezvoltat de W. Waite. Dacă BCPL și INTCODE reprezentau componenta de „mașină abstractă” și cod intermediar, STAGE-2 aducea în prim-plan macroprocesarea ca instrument de construcție de compilatoare și translatoare.

Am primit de la Waite o versiune de STAGE-2 universal pe bandă magnetică și o documentație destul de densă. Ideea centrală era fascinantă: un macroprocesor universal care să poată fi folosit pentru extinderea limbajelor, reformatări de text, generare de cod și automatizare a unor transformări complexe, toate într-un mod independent de un limbaj sau de un sistem software anume. STAGE-2 nu era gândit ca anexă la un asamblor sau la un compilator, ci ca program de sine stătător, capabil să proceseze orice flux de text și să producă altul, pe baza unui set de reguli de rescriere. Caracteristica principală a STAGE-2 este procedura de *half-bootstrap* pentru implementare portabilă. Această cercetare făcea parte dintr-un studiu mai amplu privind portabilitatea programelor.

În termeni simpli, un macroprocesor este un program care copiază un flux de text de intrare și generează un flux de text de ieșire, efectuând pe parcurs un set sistematic de înlocuiri și transformări. Procesoarele macro sunt adesea încorporate în alte programe, cum ar fi asamblatoare și compilatoare. Uneori sunt programe independente care pot fi utilizate pentru a procesa orice tip de text.

Macroprocesoarele au fost folosite, de la început, pentru:

- extinderea limbajelor, prin definirea de noi constructe exprimate în termeni de structuri existente;
- înlocuirea sistematică a fragmentelor de text în situații care implicau decizii;
- reformatarea textelor, generarea de cod repetitiv sau de schelete de programe.

STAGE-2 ducea aceste idei la un nivel de generalitate remarcabil. El procesează câte o linie de caractere, potrivind-o cu modele (*patterns*) specificate în prealabil, și este aproape complet independent de setul de caractere: are nevoie doar ca cifrele 0–9 să fie contigue și în ordine. Această independență față de setul de caractere îl transforma într-un instrument ideal pentru construirea de software independent de mașină.

La fel ca în cazul BCPL, Waite a prevăzut o implementare ușoară, pe orice computer, a macroprocesorului, folosind o procedură de *half-bootstrap* portabil: o versiune minimală de STAGE-2, suficientă pentru a interpreta o descriere simplificată a macroprocesorului, era folosită pentru a genera versiunea completă, optimizată pentru platforma concretă. Ca și în cazul BCPL, procedura lui Waite presupunea scrierea unui interpretor minim, care să poată executa o versiune simplificată a STAGE-2, suficientă cât să producă versiunea completă.

Noi am pornit de la varianta generică primită de la Waite și, împreună cu câțiva studenți foarte buni, am realizat pe FELIX această procedură de *half-bootstrap*: un „nucleu” STAGE-2 simplificat, folosit pentru a genera STAGE-2 complet.

Această experiență a fost importantă din două motive:

- tehnic, demonstra că portabilitatea nu este o proprietate pasivă, ci o construcție activă, care cere viziune, adaptabilitate și rigoare;
- metodologic, arăta cum un singur instrument — un macroprocesor universal — poate deveni infrastructură pentru multe alte unelte: compilatoare, traductoare, generatoare de cod.

Articol privind experiența dobândită din cercetarea portabilității cu macroprocesorul STAGE -2:

Specialized Macroprocessor for Software Development – L. D. Șerbănați, R. D. Bercaru, D. F. Mânduțianu, *Automatics, Metrology, Computers*, Vol. 27, pp. 103–115, Technical Publishing House, Bucharest, 1978 (în română).

Plecând de la experiența cu STAGE-2, am propus aici un macroprocesor specializat pentru dezvoltarea de software. Ideea era să construim, deasupra unui macroprocesor general, un set de facilități dedicate: șabloane pentru structurile tipice ale compilatoarelor, descrieri de tabele de simboluri, generatoare de cod parametrizabile. Scopul era să obținem o infrastructură care să facă mai ușoară și mai sistematică dezvoltarea de unelte portabile.

Dacă STAGE-2 reprezenta, în viziunea lui Waite, un macroprocesor universal, acest articol marchează o derivare spre un instrument specializat pentru dezvoltarea de software. Experiența cu STAGE-2, cu tehnicile de *bootstrap* și cu ideea de a construi software independent de mașină m-a condus la întrebarea: putem avea un macroprocesor adaptat explicit nevoilor procesului de dezvoltare, nu doar un instrument general de prelucrare de text?

În lucrare am descris proiectarea și implementarea unui macroprocesor specializat care:

- accepta descrieri orientate pe structuri tipice ale programelor (fragmente de cod, șabloane de modul, convenții de apel);
- integra, în măsura posibilului, experiența acumulată în laborator — de exemplu, patternuri uzuale de generare de cod portabil;
- putea fi folosit ca infrastructură pentru alte unelte, de la generatoare de cod până la instrumente de configurare a mediului de dezvoltare.

În fundal, se simte deja ecoul ideilor lui Strachey despre macrogeneratorul universal GPM și al visului, mai vechi, UNCOL — un limbaj intermediar universal pentru compilatoare. Articolul nostru aducea această linie de gândire în contextul concret al informaticii românești, cu resurse limitate, dar cu ambiția de a lucra în același orizont conceptual.

STAGE-2 a fost, pentru mine, o punte între experiența de portabilitate legată de compilatoare și preocupările viitoare pentru modelarea procesului de proiectare: ideea de a avea un instrument care „transformă texte” în mod sistematic, pe baza unor reguli explicite, anticipa, într-un fel, nevoia de a trata proiectarea ca o succesiune de transformări controlate asupra unor forme intermediare.

Implementările portabile ale compilatorului limbajului BCPL și ale macroprocesorului STAGE-2 le-am dezvoltat împreună cu studenții. În jurul BCPL, INTCODE și STAGE-2 am format, treptat, un mic colectiv dedicat portabilității limbajelor și programelor, din care făceau parte colegi ca Maria Ana Necula, Irina Athanasie, și studenți ca M. Cosma, C. Popescu, R. D. Bercaru, D. F. Mânduțianu și alții. Munca din laborator — scrierea interpretorului INTCODE, *bootstrap*-ul compilatorului BCPL, implementarea STAGE-2 pe FELIX — s-a transformat treptat în articole care au încercat să structureze experiența noastră într-un limbaj teoretic accesibil comunității. Ele nu au fost simple exerciții tehnice, ci manifestări ale unei idei profunde: că un program, asemenea unei idei umane, ar trebui să poată supraviețui și evolua dincolo de circumstanțele nașterii sale.

Janus

O experiență suplimentară cu portabilitatea mi-a fost oferită de un alt macroprocesor propus de Waite, ultimul în ordine cronologică: implementarea lui Janus — un limbaj simbolic folosit pentru a îngloba informațiile care sunt, în mod normal, transmise de la faza de analiză a unui compilator la generatoarele de cod. Spre deosebire de STAGE-2 care utilizează o mașină abstractă pentru a implementa un singur limbaj sursă pe o varietate de arhitecturi țintă, Janus se apropie de ideea originală a lui UNCOL prin definirea elementelor esențiale ale unui limbaj intermediar independent de

orice limbaj sursă sau arhitectură țintă. În loc să ofere o singură mașină abstractă, Janus descrie structura de bază comună (mașină stivă cu registru index) a unei întregi familii de mașini abstracte, care diferă prin seturile lor de instrucțiuni pentru a se adapta construcțiilor specifice ale limbajelor sursă.

Janus era, deci, conceput pentru „transportul” de software: un program codificat într-un limbaj de nivel înalt poate fi tradus de Janus pe un computer, iar rezultatul poate fi tradus în cod de asamblare pe altul (macroprocesorul STAGE-2 urma să fie utilizat pentru a doua traducere). Această temă i-am propus-o lui Dan Mânduțianu ca subiect pentru lucrarea lui de diplomă. Dan a dus la bun sfârșit implementarea, dar, din păcate, studiile noastre s-au oprit aici, din cauza altor proiecte apărute la orizont.

Aceste experiențe m-au modelat nu doar ca informatician, ci și ca gânditor: m-au învățat să nu privesc limitele prezente ca frontiere definitive, ci ca invitații la creație. Eliberarea software-ului de cătușele hardware-ului și ale sistemelor de operare nu a fost, în fond, decât o expresie a eliberării gândirii — un prim pas pe un drum lung, în care programarea nu mai era doar o tehnică, ci o artă a eliberării de restricții.