L. FREVERT

Gesellschaft für Kernforschung mbH.
Projekt PDV

Prozeßprogrammiersprache PEARL

Prozeßprogrammiersprache PEARL

(Vortrag auf Tagung des Siemens-Anwenderkreises I vom 11.-13. 4. 1973)

Entwurfsziele

PEARL ist eine Gemeinschaftsentwicklung von Mitarbeitern aus Industriefirmen und Forschungsinstituten. Der Name steht als Abkürzung für "Process and Experiment Automation Realtime Language".
PEARL ist eine methodenorientierte Sprache, in der sich alle Aufgaben der Prozeßautomatisierung formulieren lassen sollten. Bei ihrer Entwicklung wurden folgende Ziele angestrebt:

1) Weite Anwendbarkeit
2) Übertragbarkeit der Programme auf andere Rechner
3) Sicherheit
4) Leichte Erlernbarkeit

PEARL soll eine Sprache sein, die auch von relativ ungeübten Benutzern zur Programmierung von Prozeßautomatisierungsanlagen verwendet werden kann. PEARL soll nicht als Systemprogrammiersprache dienen (Abgrenzung gegenüber Sprachen wie POLYP). Deshalb fehlen in PEARL Sprachelemente (z.B. für Adreßrechnung), die die Gefahr schwer erkennbarer Fehler in sich bergen und geübte Programmierer voraussetzen. Programmierfehler sollten möglichst schon beim Compilieren erkannt werden können; deshalb muß vieles in den Programmen redundant formuliert werden; Sicherheit wird mit höherer Schreibarbeit erkauft (keine impliziten Deklarationen).

Eigenschaften von PEARL

(Siehe auch Vortrag von Herrn Elzer im Tagungsbericht des Siemens-Anwenderkreises I vom 15. - 17.3. 1972 in Paderborn)
PEARL entspricht im algorithmischen Teil weitgehend den bekannten Sprachen wie PL/1 oder ALGOL.
Stichworte: Blockstruktur (dynamische Felder), Erweiterbarkeit (Deklaration neuer Datentypen und Operatoren), Strukturen (Felder aus Daten unterschiedlichen Typs).

Als Prozeßprogrammiersprache bietet PEARL selbstverständlich alle wünschbaren Möglichkeiten zum Hantieren mit Bits und Bitketten. Indirekte Adressierung ist über ein zweistufiges Referenzkonzept möglich, jedoch aus Sicherheitsgründen keine Adressrechnung.
Der Ein/Ausgabeteil wurde gegenüber konventionellen Sprachen wesentlich erweitert um graphische Ein/Ausgabe und um nichtformatierten Datentransfer (Transfer von Bitketten, evtl. unter Anwendung von Eichtransformationen).

Gegenüber konventionellen Sprachen hinzugekommen sind die Möglichkeiten, Echtzeitaufgaben zu lösen, das heißt, Parallelarbeit zu formulieren und durch Ereignisse im Prozeß (z.B. durch Interrupts) anstoßen zu lassen.
Derartige Aufgaben werden heute bei der Prozeßprogrammierung durch Aufrufe an das Betriebssystem gelöst (ORG-Aufrufe). Bei PEARL ist die Schnittstelle zum Betriebssystem standardisiert worden und Element der Sprache. Das bedeutet, daß das Betriebssystem eines PEARL-Rechners in seinen wesentlichen Eigenschaften durch die Sprache festgelegt ist; im allgemeinen gehört zur Implementierung von PEARL das Schreiben des Betriebssystems, zumindest eine Anpassung des vorhandenen. Hier unterscheidet sich PEARL auch von Prozeßprogrammiersprachen wie CORAL.
Ein Beispiel soll einen Eindruck von den Möglichkeiten geben:

```
ON STARTSIGNAL AFTER 3 MIN ALL 3Ø SEC DURING 5.5 MIN
     ACTIVATE MASSNAHME PRIORITY 5;
```

MASSNAHME ist eine an anderer Stelle deklarierte Task (sequentieller Teilprozeß).
Selbstverständlich können parallele Tasks auch von anderen Tasks angestoßen, gestoppt, wiedergestartet oder abgebrochen werden.
Für die Organisation von Parallelarbeit unerläßlich sind Möglichkeiten, Parallelarbeit zu synchronisieren. Z.B. muß häufig verhindert werden, daß zwei Tasks gleichzeitig auf ein Datum zugreifen und es zu ändern versuchen. Diese Aufgaben lassen sich in PEARL durch Semaphoroperationen (nach Dijkstra) und Bolt-Operationen lösen. (Die Semaphoroperationen entsprechen ungefähr dem Belegen und Freigeben im ORG).

Die Standardisierung der Schnittstelle zwischen Betriebssystem und Benutzerprogramm ermöglicht die Übertragung von PEARL-Programmen auf Rechner anderen Typs. Selbstverständlich werden dieser Übertragbarkeit Grenzen gesetzt durch unterschiedliche Prozeßperipherien. Dem ist bei PEARL dadurch Rechnung getragen, daß alle vom jeweiligen Prozeßinstrumentierungssystem abhängigen Dinge in einem besonderen Systemteil der Programme zusammengefaßt beschrieben werden müssen, während das von der

Instrumentierung unabhängige Prozeßlenkungsproblem im Problemteil programmiert wird. Beim Übergang auf eine andere Anlage oder bei Änderung der Peripherie wird nur der Systemteil geändert.

Fragen der Effizienz

Es wird sicher nach der Effizienz von Programmen gefragt werden, die in PEARL geschrieben werden. Dazu nur folgendes:

1) Prozeßlenkungsprogramme kosten bei Assemblerprogrammierung ca. 60 DM/Befehl (Mannjahr 100.000 DM, 1 Befehl/Stunde einschl. Prozeßanalyse und Dokumentation). 1 Kernspeicherwort kostet etwa 5 DM. Wenn die Programmierkosten durch PEARL um 20% gesenkt werden können, ergibt sich der gleiche Endpreis, wenn die dabei entstehenden Objektprogramme dreimal so lang sind wie bei Assemblerprogrammierung. Diese Rechnung ist natürlich nur vernünftig, wenn das längere Programm die vom Prozeß gesetzten Zeitgrenzen nicht überschreitet.

2) Umfangreiche Prozeßlenkungsprogramme, die in Assembler geschrieben sind, laufen oft jahrelang ohne Änderung, weil sich kein Mensch traut, an und für sich wünschenswerte Änderungen vorzunehmen. Daraus wird dann häufig geschlossen, daß man ruhig in Assembler programmieren kann, weil die Programme erfahrungsgemäß ja doch nie geändert werden.

Zu lösende Probleme

Selbstverständlich benötigt ein Rechner, der alle Echtzeitmöglichkeiten von PEARL erfüllen soll, ein sehr unfangreiches Betriebssystem. Das ist aber kein Argument gegen PEARL, sondern ein Argument dafür, nach Verfahren zu suchen, die es gestatten, das Betriebssystem eines Rechners an eine bestimmte Prozeßlenkungsaufgabe zu adaptieren, so daß es nur die Funktionen erfüllt, die wirklich im Programm gebraucht werden. Die heutige Technik, Betriebssysteme aus Komponenten zusammenzusetzen, läßt hier viel zu wünschen übrig.

Fragen der Effizienz sollten durch Weiterentwicklung von Methoden zur automatischen Optimierung gelöst werden. Die Programmierer sollten sich auf die Lösung der Prozeßlenkungsaufgaben und nicht auf Tricks konzentrieren müssen. Im übrigen sind von Hand optimierte Programme schwerer zu durchschauen, als nicht optimierte.

Ein sehr wesentliches Problem bei der Programmierung von Echtzeitaufgaben stellt die Vermeidung von Programmierfehlern dar, die zu Verklemmungen und fehlerhaften Synchronisationen führen. Beispiel: Zwei parallel laufende Teilprogramme belegen Kartenleser und Drucker in verschiedener Reihenfolge. Wenn beide zufällig jeweils das erste Gerät gleichzeitig belegen, warten beide auf die Freigabe des jeweils zweiten (das erste Programm belegt zuerst den Kartenleser, das zweite Programm gleichzeitig den Drucker; das erste möchte den Drucker, bekommt ihn aber nie, weil das zweite ihn hat und auf den Kartenleser wartet).

Im Rechenzentrumsbetrieb ist man heute zufrieden, wenn das Echtzeitsystem nur einmal täglich zusammenbricht; bei Prozeßlenkungen ist eine solche Fehlerrate indiskutabel. Durch eine systematische Untersuchung der Programme sind theoretisch derartige mögliche Fehler erkennbar. Solche Untersuchungen übersteigen die menschlichen Fähigkeiten, können aber prinzipiell von Rechnern durchgeführt werden. Es fehlen lediglich die entsprechenden Algorithmen. Die Anwendung derartiger Prüfalgorithmen setzt jedoch voraus, daß die Echtzeitprobleme in einer höheren Programmiersprache formuliert sind. Es kann vermutet werden, daß dies später eines der wesentlichsten Argumente für die Anwendung von PEARL oder ähnlichen Sprachen für Echtzeitprogrammierung bilden wird.

Stand der Implementierungen

Gegenwärtig arbeiten die Firmen AEG-Telefunken, BBC, Siemens und eine Arbeitsgemeinschaft aus Universitätsinstituten in Stuttgart und Erlangen und der Münchner Systemfirma ESG an PEARL-Implementierungen. Implementiert wird für AEG 60-50, PDP-11, S306 und S330 und für einen neuen AEG-Rechner. Der erste Compiler soll noch in diesem Jahre laufen. Besonders interessant ist die Implementation der Arbeitsgemeinschaft, die ein möglichst portables Prozeßprogrammiersystem auf der Grundlage von PEARL entwickeln will.
Selbstverständlich arbeiten alle Implementationsgruppen eng zusammen, um die Sprachbeschreibung einheitlich zu interpretieren und damit die Übertragbarkeit der PEARL-Programme zu gewährleisten. Es ist unser Ziel, möglichst große Teile von PEARL in eine international standardisierte Prozeßrechnersprache einfließen zu lassen.

Als flankierende Maßnahme werden im Projekt PDV Verfahren zur adaptiven Betriebssystemgenerierung entwickelt. Ein allgemeines PEARL-Betriebssystem wird in seinen rechnerunabhängigen Teilen in einer höheren Programmiersprache geschrieben und hierfür Optimierungsalgorithmen und Algorithmen zur Prüfung von Echtzeitprogrammen entwickelt. Wir hoffen, daß durch diese Maßnahmen bis zum Auslaufen des 2. DV-Programmes die Prozeßprogrammierung mit PEARL eine gesicherte Technik darstellen wird.