Fortran 95-Syntax
Eisenbahnschienen-Diagramme

Die formale Beschreibung der Syntax von Fortran 95 besteht aus BNF–Regeln. Der vorliegende Umdruck beschreibt die Syntax von Fortran 95 anhand von Eisenbahnschienen-Diagrammen. Als Vorlage für diese Diagramme dienten die BNF–Regeln im Entwurf des Standarddokumentes „Committee Draft, March 1996“ für Fortran 95.

Diese Diagramme sind keine 100%ig äquivalente Darstellung der Syntax, sie sind jedoch meistens leichter zu lesen. Bestimmte grammatische Eigenschaften können mit solchen Diagrammen nicht dargestellt werden. Dazu gehören

1. die Art und Weise, wie man eine Anweisung auf eine oder mehr Zeilen fortsetzen kann,
2. die Art und Weise, wie eine Anweisung einer anderen Anweisung folgt,
3. Kommentarzeilen und
4. kontextabhängige und sonstige Eigenschaften, die im Standarddokument in Form von Zusatzbedingungen formuliert sind.

Obwohl der Umdruck nach bestem Wissen erstellt wurde, übernimmt das RRZN keine Garantie für seine Korrektheit.

1 Konventionen

Großbuchstaben und Sonderzeichen des Fortran-Zeichensatzes sind in `dieser Schrift` dargestellt und müssen meistens genauso geschrieben werden. Kleinbuchstaben und kleingeschriebene Wörter (oft mit Bindestrich und mit bestimmten Abkürzungen), die in *dieser Schrift* dargestellt sind, bezeichnen Metavariablen, die in tatsächlichen Anweisungen ersetzt werden müssen.

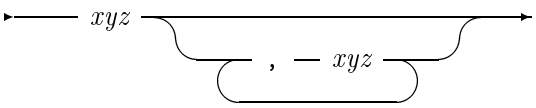
Innerhalb der Bezeichnung vieler Metavariablen werden Abkürzungen verwendet:

stmt	für	Anweisung	*attr*	für	Attribut
expr	für	Ausdruck	*decl*	für	Vereinbarung
op	für	Operator	*desc*	für	Format
int	für	ganzzahlig(er, es)	*spec*	für	Spezifizierer; hier
arg	für	Parameter			Angabe (in Spezifikationen) bzw.
def	für	Definition			Parameter (in E/A-Anweisungen)

Jedes Diagramm hat eine eindeutige Nummer der Form R*snn*, wobei *s* eine ein- oder zweistellige Kapitelnummer innerhalb des Standarddokumentes ist und *nn* eine zweistellige Folgenummer innerhalb des Kapitels ist. Unvollständige Diagramme, die erst später in anderen Kapiteln in allen Details erläutert werden, erkennt man daran, daß ihre Nummer nicht zum laufenden Kapitel paßt.

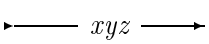
Um die Anzahl der Diagramme insgesamt gering zu halten, werden folgende Diagramme vorausgesetzt:

xyz-list



xyz-name

scalar-xyz

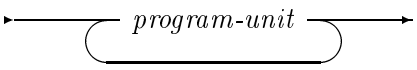


Weitere nichtdefinierte Metavariablen sind *letter*, *digit*, *special-character* und *rep-char*.

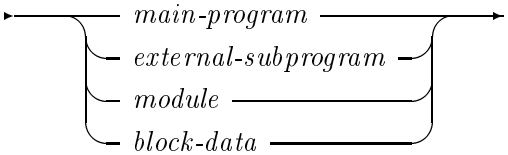
Die folgende Einteilung der Diagramme einschließlich der Kapitelüberschriften entspricht dem Standarddokument.

2 Begriffe und Konzepte

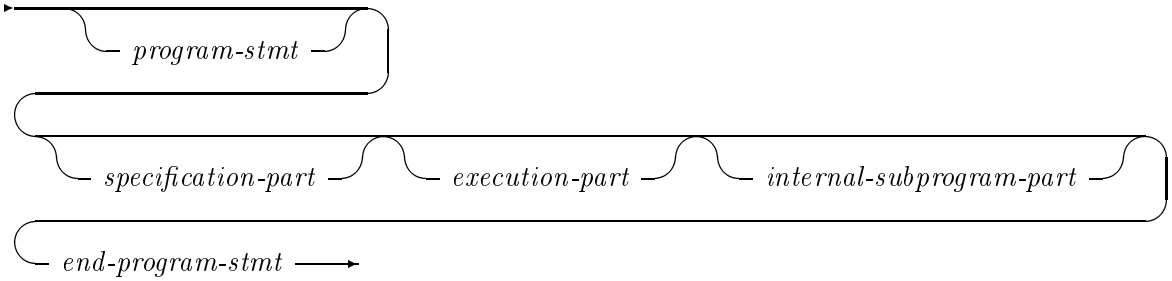
R201 *program*



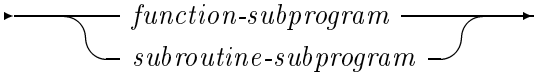
R202 *program-unit*



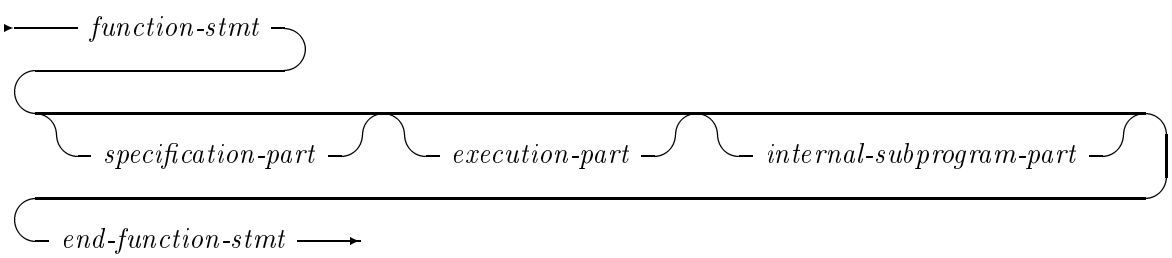
R1101 *main-program*



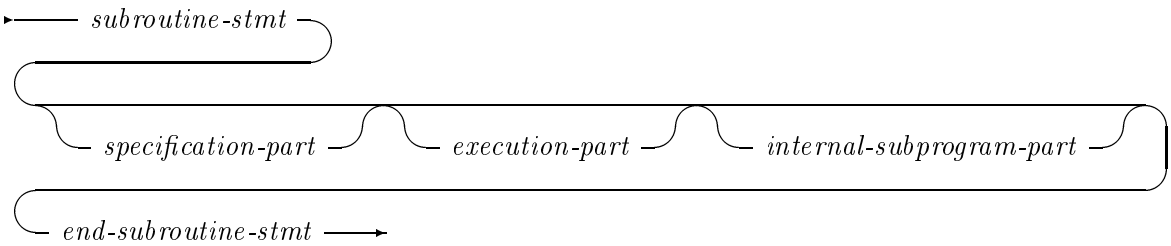
R203 *external-subprogram*



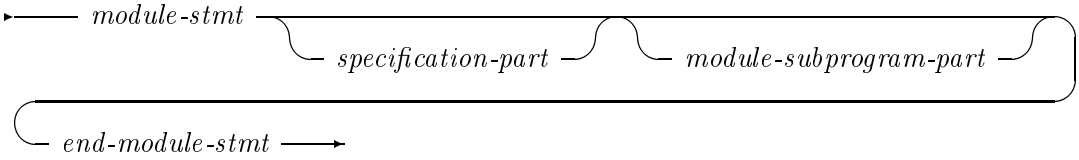
R1216 *function-subprogram*



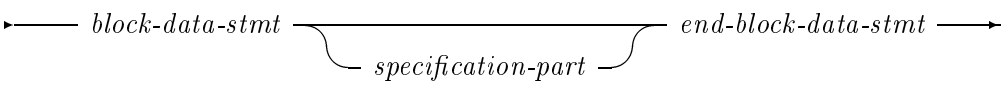
R1221 *subroutine-subprogram*



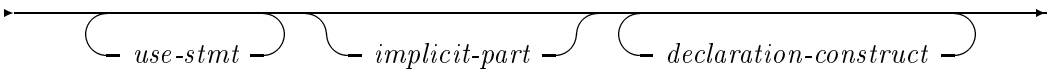
R1104 *module*



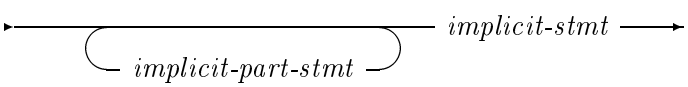
R1112 *block-data*



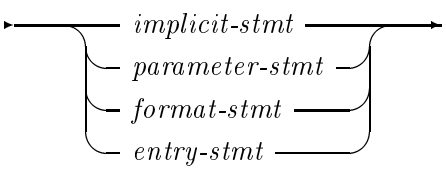
R204 *specification-part*



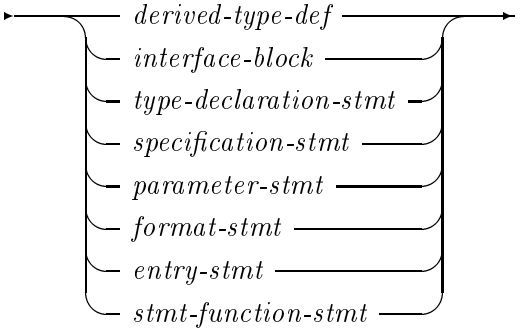
R205 *implicit-part*



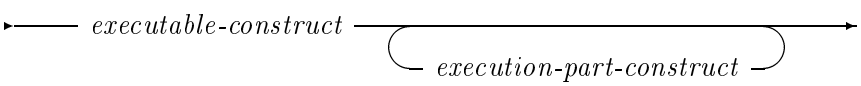
R206 *implicit-part-stmt*



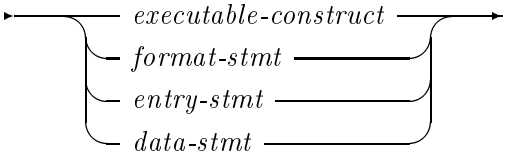
R207 *declaration-construct*



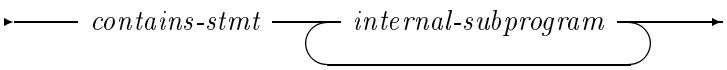
R208 *execution-part*



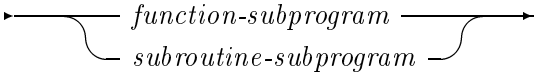
R209 *execution-part-construct*



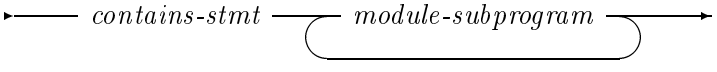
R210 *internal-subprogram-part*



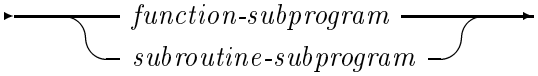
R211 *internal-subprogram*



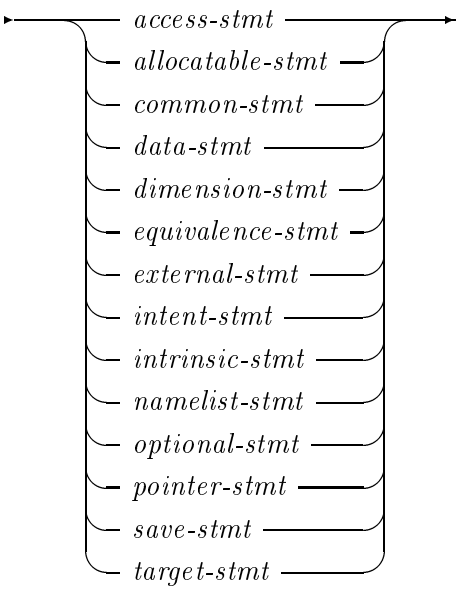
R212 *module-subprogram-part*



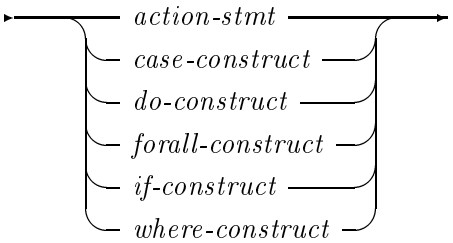
R213 *module-subprogram*



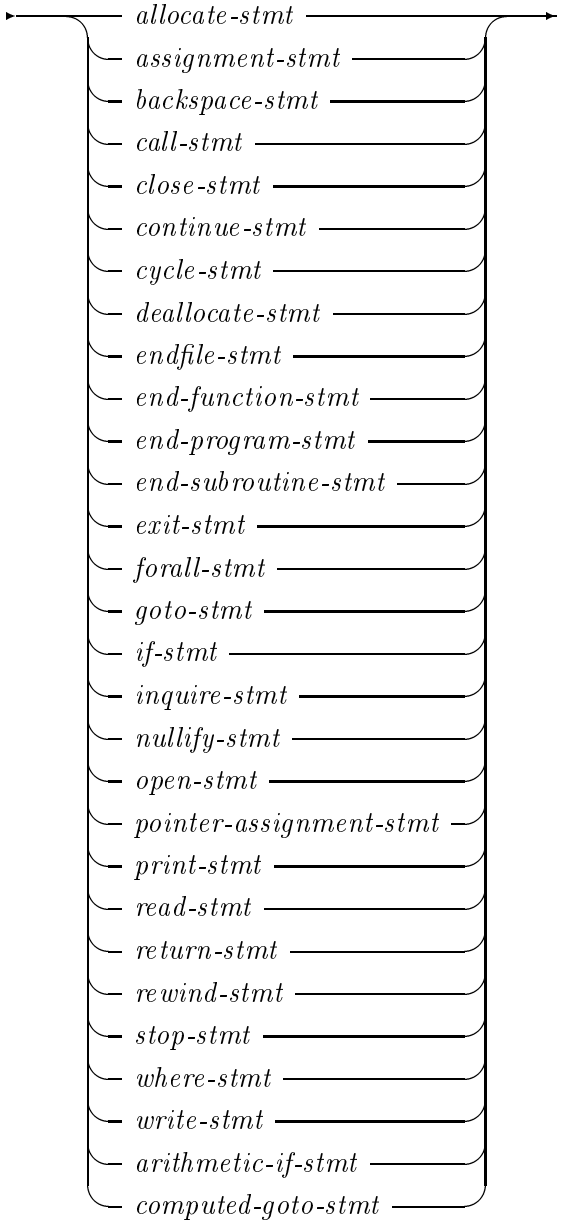
R213 *specification-stmt*



R215 *executable-construct*

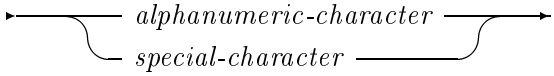


R216 *action-stmt*

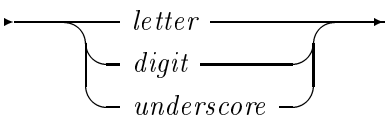


3 Zeichen, lexikalische Grundelemente, Form des Quelltextes

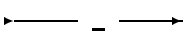
R301 *character*



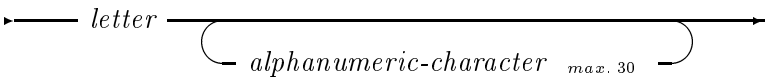
R302 *alphanumeric-character*



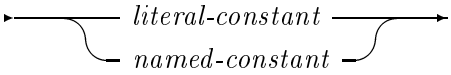
R303 *underscore*



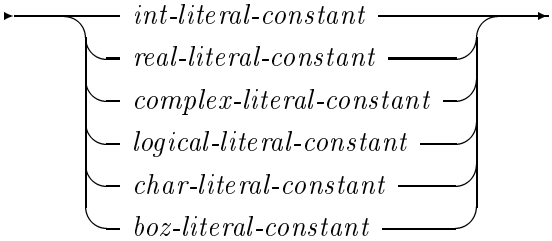
R304 *name*



R305 *constant*



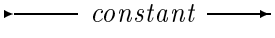
R306 *literal-constant*



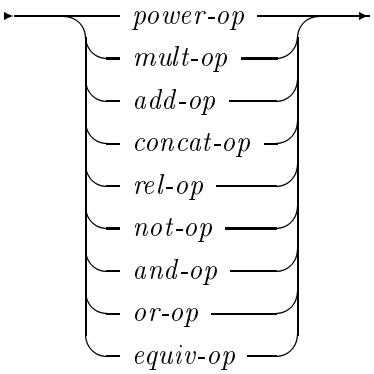
R307 *named-constant*

R308 *int-constant*

R309 *char-constant*

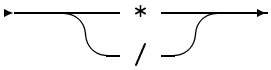


R310 *intrinsic-operator*

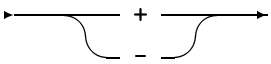


R708 *power-op*

R709 *mult-op*

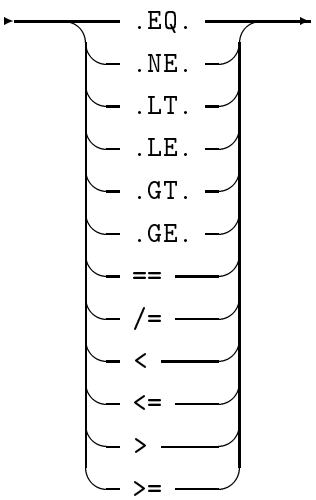


R710 *add-op*



R712 *concat-op*

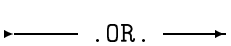
R714 *rel-op*



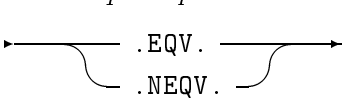
R719 *not-op*

R720 *and-op*

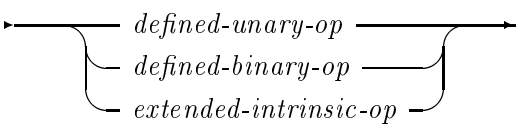
R721 *or-op*



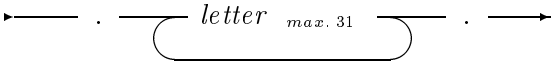
R722 *equiv-op*



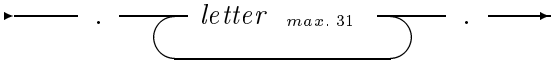
R311 *defined-operator*



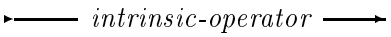
R704 *defined-unary-op*



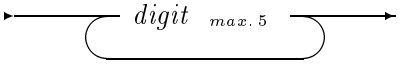
R724 *defined-binary-op*



R312 *extended-intrinsic-op*

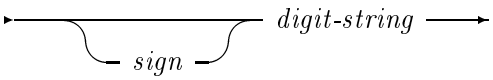


R313 *label*

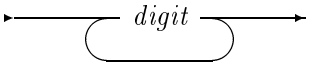


4 Vordefinierte und benutzerdefinierte Datentypen

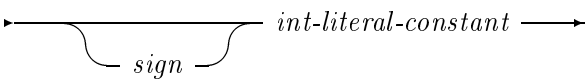
R401 *signed-digit-string*



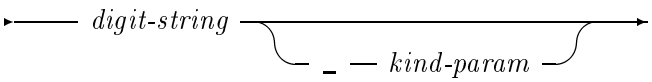
R402 *digit-string*



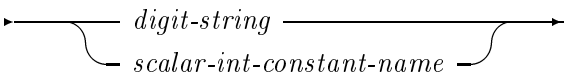
R403 *signed-int-literal-constant*



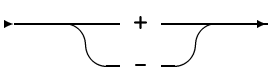
R404 *int-literal-constant*



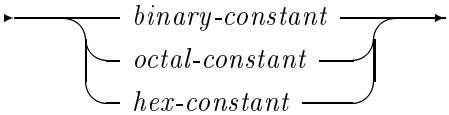
R405 *kind-param*



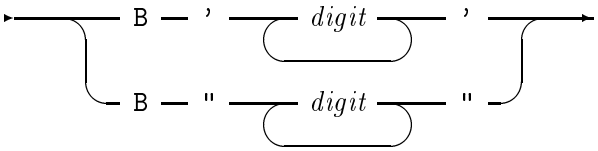
R406 *sign*



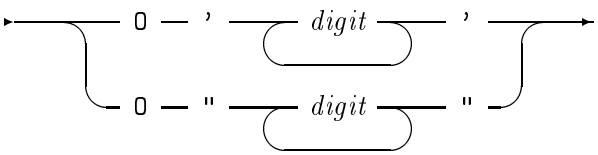
R407 *boz-literal-constant*



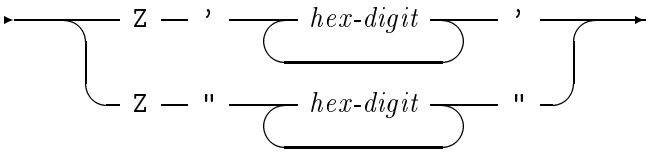
R408 *binary-constant*



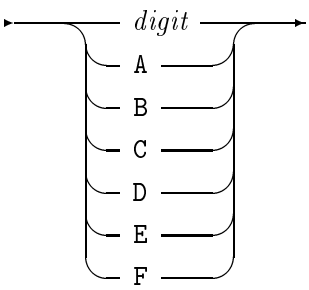
R409 *octal-constant*



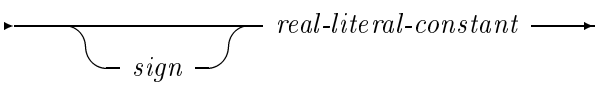
R410 *hex-constant*



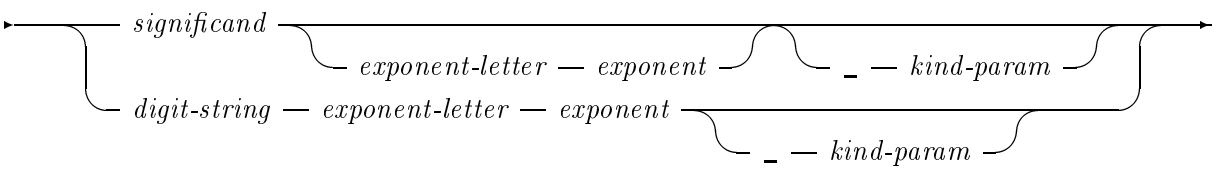
R411 *hex-digit*



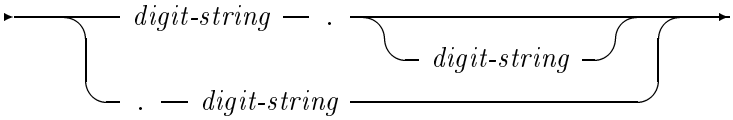
R412 *signed-real-literal-constant*



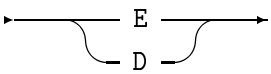
R413 *real-literal-constant*



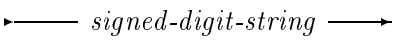
R414 *significand*



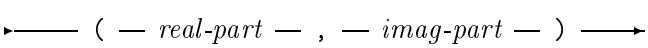
R415 *exponent-letter*



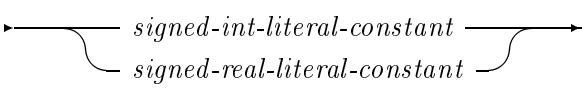
R416 *exponent*



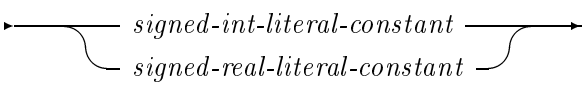
R417 *complex-literal-constant*



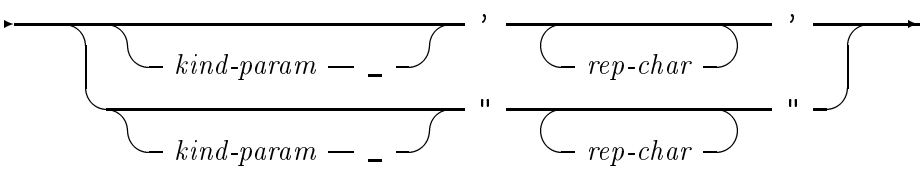
R418 *real-part*



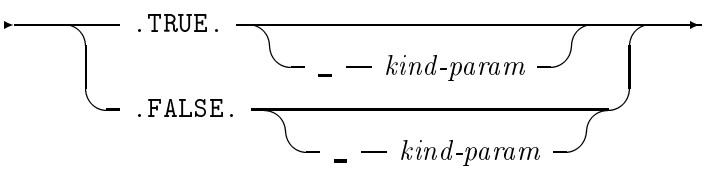
R419 *imag-part*



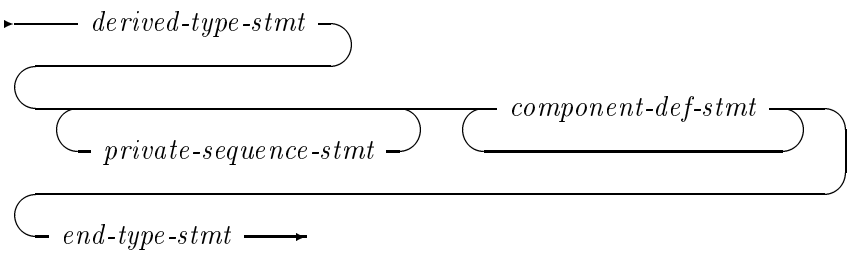
R420 *char-literal-constant*



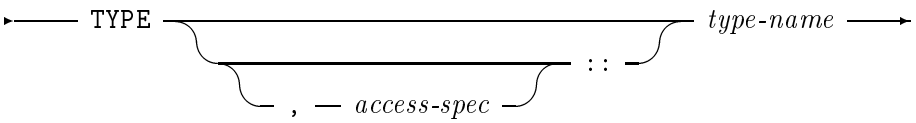
R421 *logical-literal-constant*



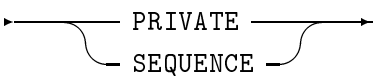
R422 *derived-type-def*



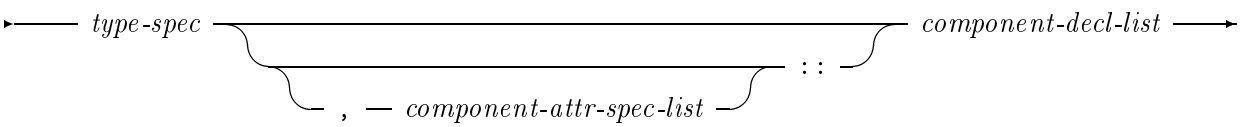
R423 *derived-type-stmt*



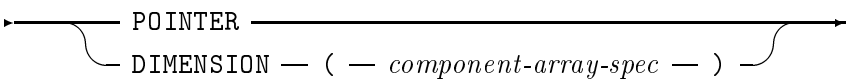
R424 *private-sequence-stmt*



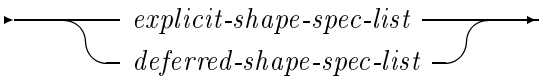
R425 *component-def-stmt*



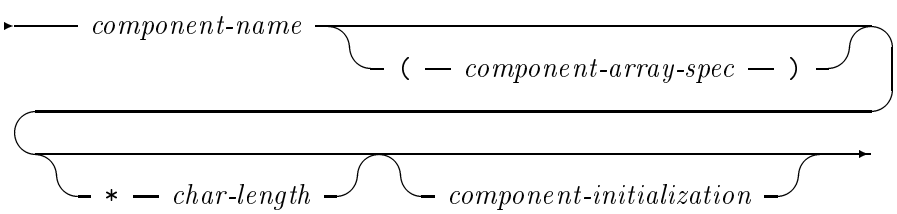
R426 *component-attr-spec*



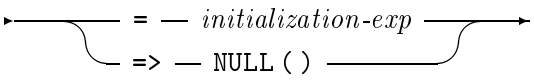
R427 *component-array-spec*



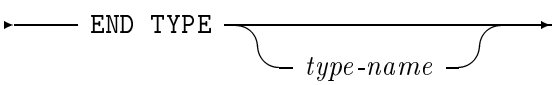
R428 *component-decl*



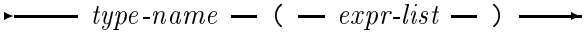
R429 *component-initialization*



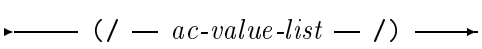
R430 *end-type-stmt*



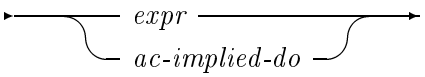
R431 *structure-constructor*



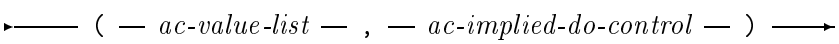
R432 *array-constructor*



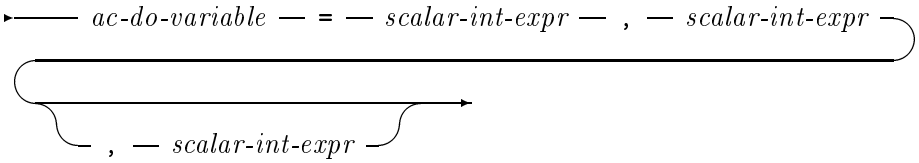
R433 *ac-value*



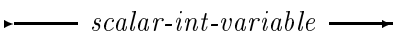
R434 *ac-implied-do*



R435 *ac-implied-do-control*

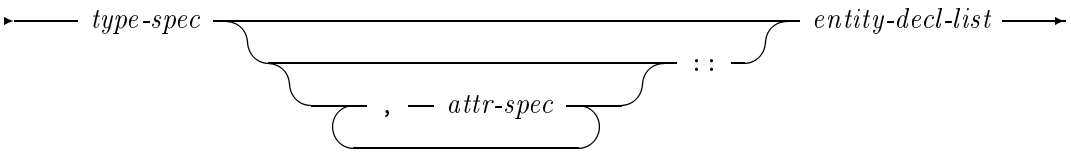


R436 *ac-do-variable*

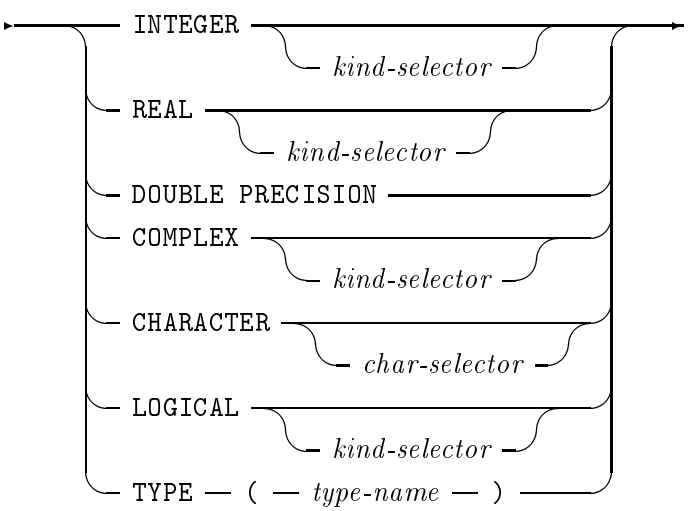


5 Vereinbarungen von Datenobjekten und Spezifikationen

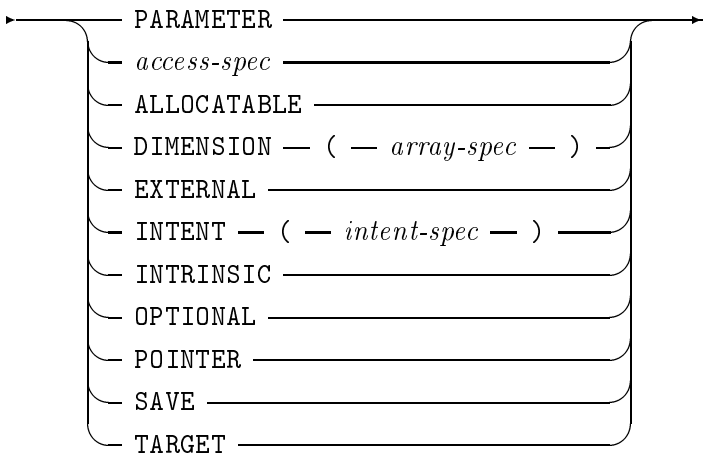
R501 *type-declaration-stmt*



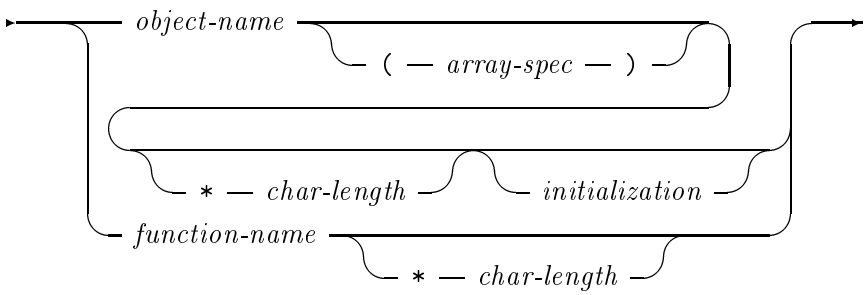
R502 *type-spec*



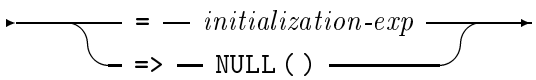
R503 *attr-spec*



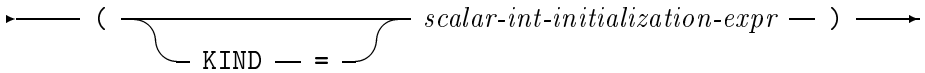
R504 *entity-decl*



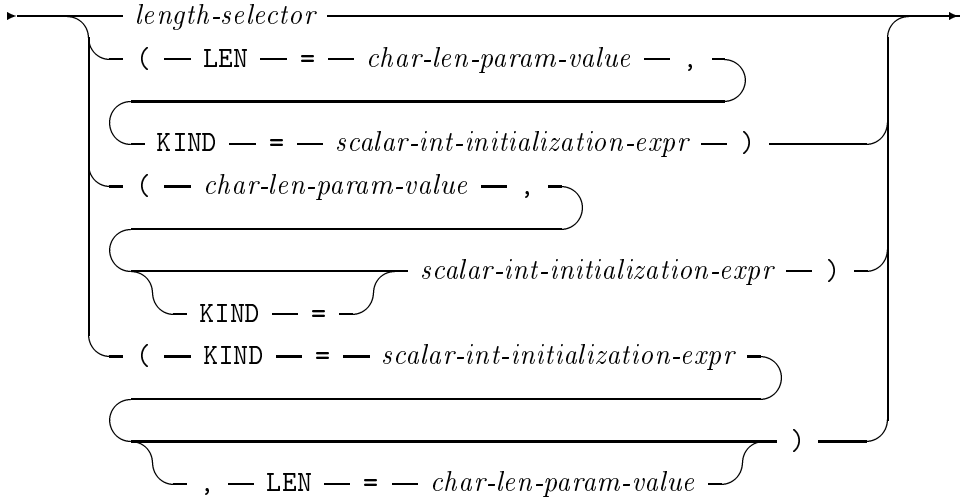
R505 *initialization*



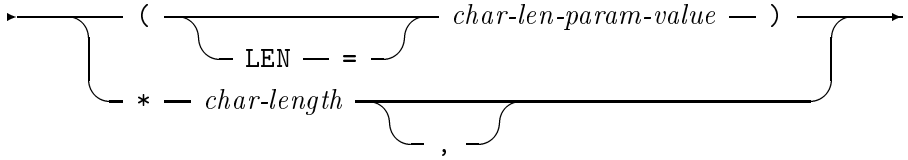
R506 *kind-selector*



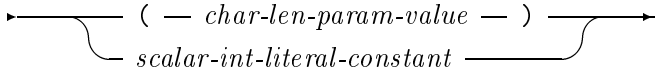
R507 *char-selector*



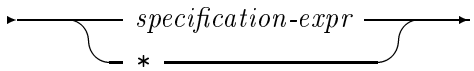
R508 *length-selector*



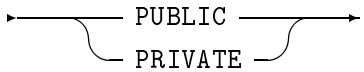
R509 *char-length*



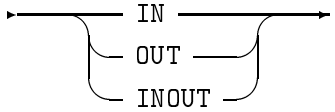
R510 *char-len-param-value*



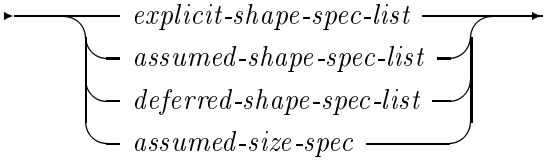
R511 *access-spec*



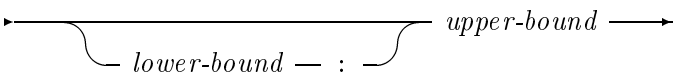
R512 *intent-spec*



R513 *array-spec*



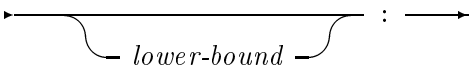
R514 *explicit-shape-spec*



R515 *lower-bound*

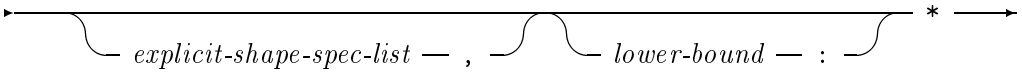
R516 *upper-bound*

R517 *assumed-shape-spec*

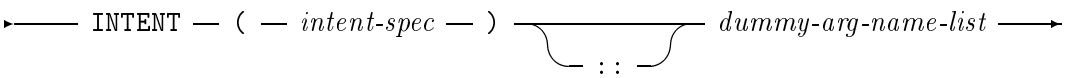


R518 *deferred-shape-spec*

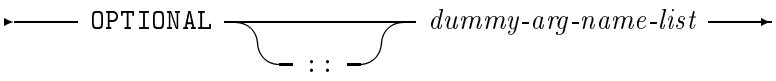
R519 *assumed-size-spec*



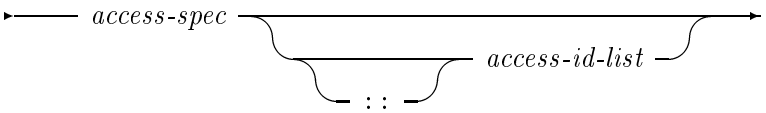
R520 *intent-stmt*



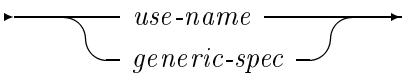
R521 *optional-stmt*



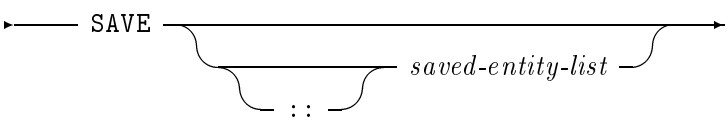
R522 *access-stmt*



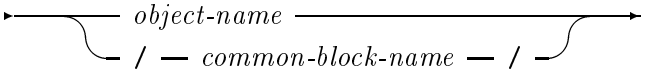
R523 *access-id*



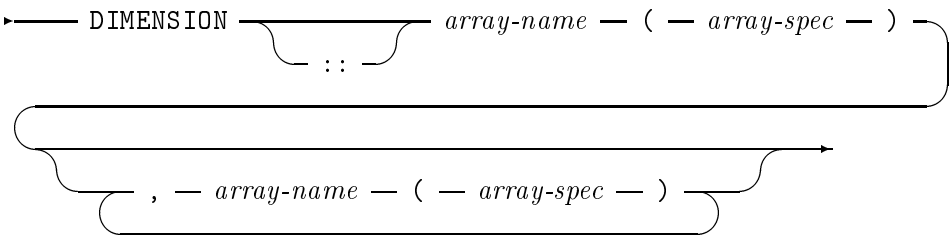
R524 *save-stmt*



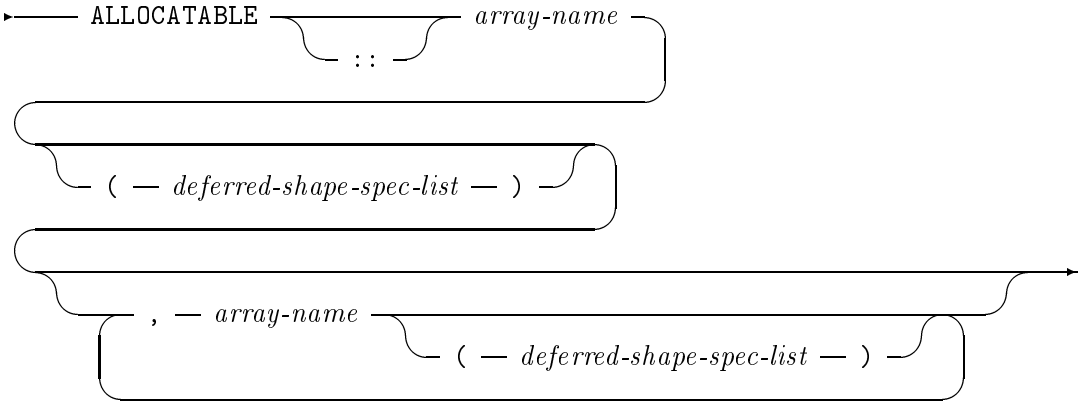
R525 *saved-entity*



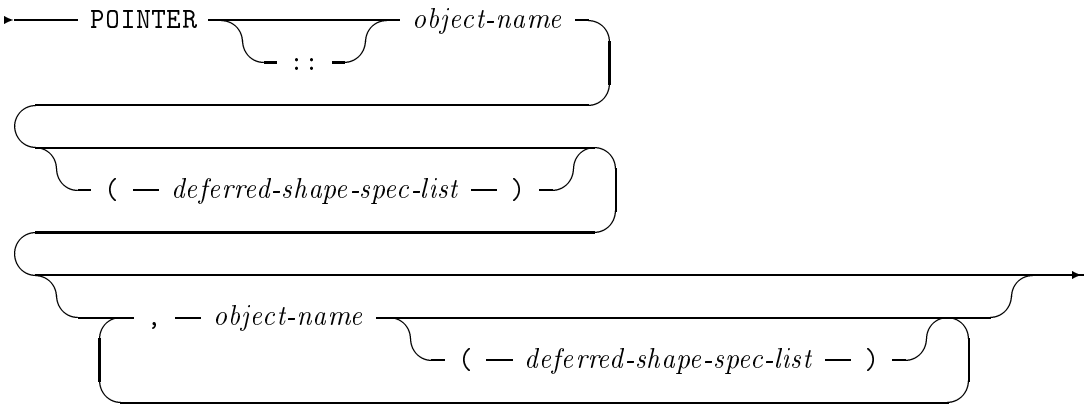
R526 *dimension-stmt*



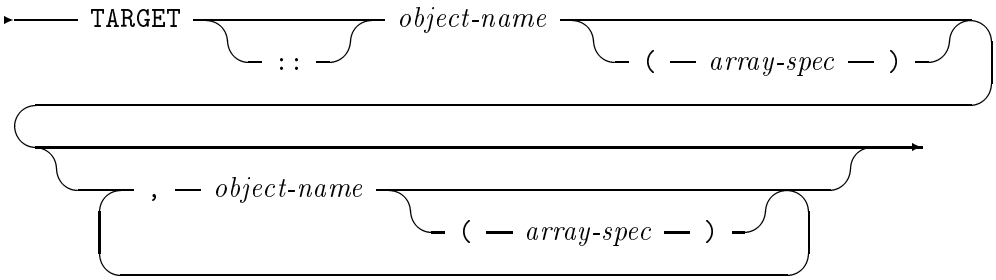
R527 *allocatable-stmt*



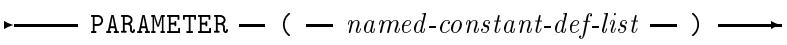
R528 *pointer-stmt*



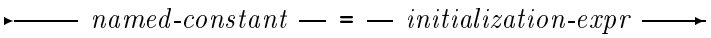
R529 *target-stmt*



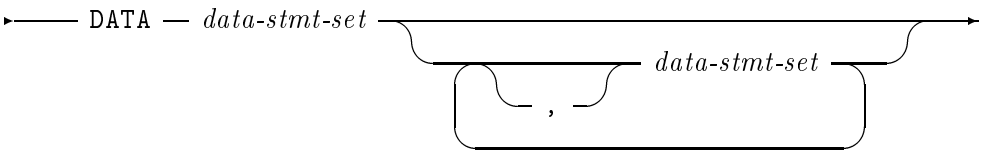
R530 *parameter-stmt*



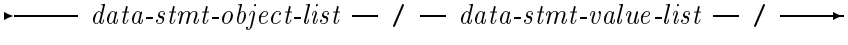
R531 *named-constant-def*



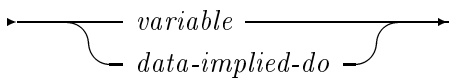
R532 *data-stmt*



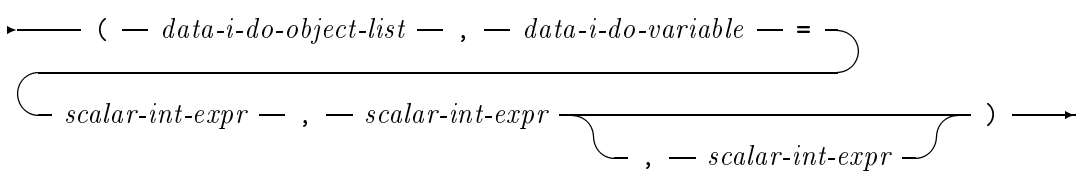
R533 *data-stmt-set*



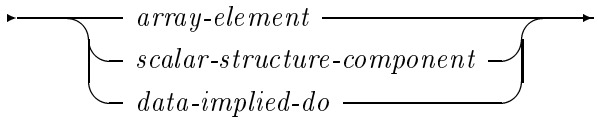
R534 *data-stmt-object*



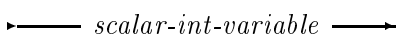
R535 *data-implied-do*



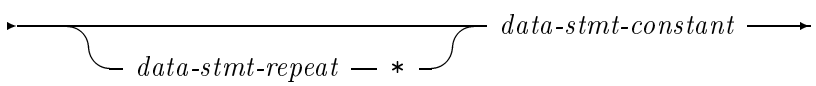
R536 *data-i-do-object*



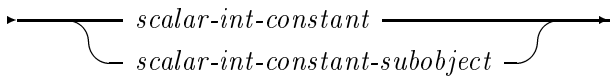
R537 *data-i-do-variable*



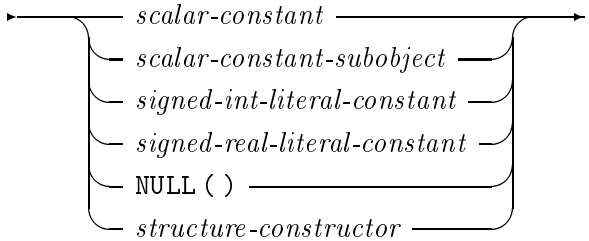
R538 *data-stmt-value*



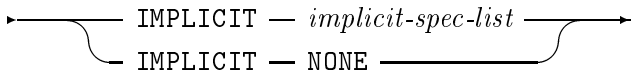
R539 *data-stmt-repeat*



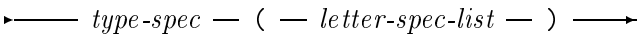
R540 *data-stmt-constant*



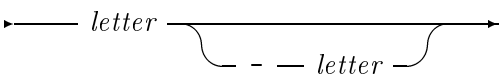
R541 *implicit-stmt*



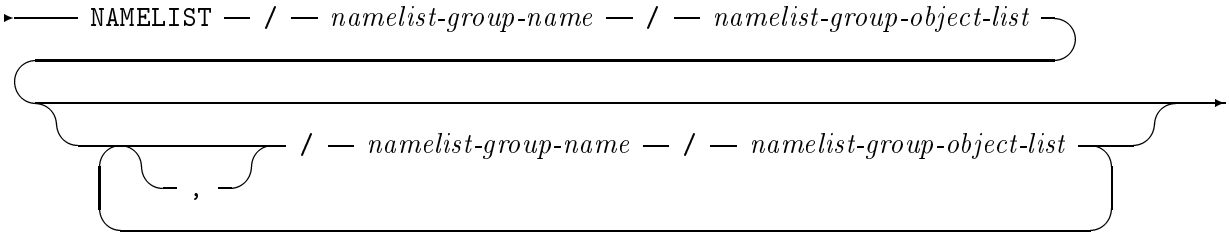
R542 *implicit-spec*



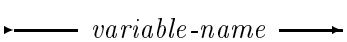
R543 *letter-spec*



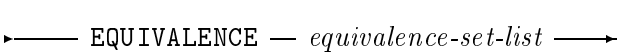
R544 *namelist-stmt*



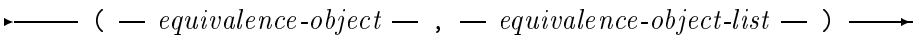
R545 *namelist-group-object*



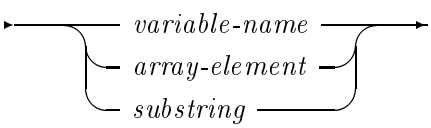
R546 *equivalence-stmt*



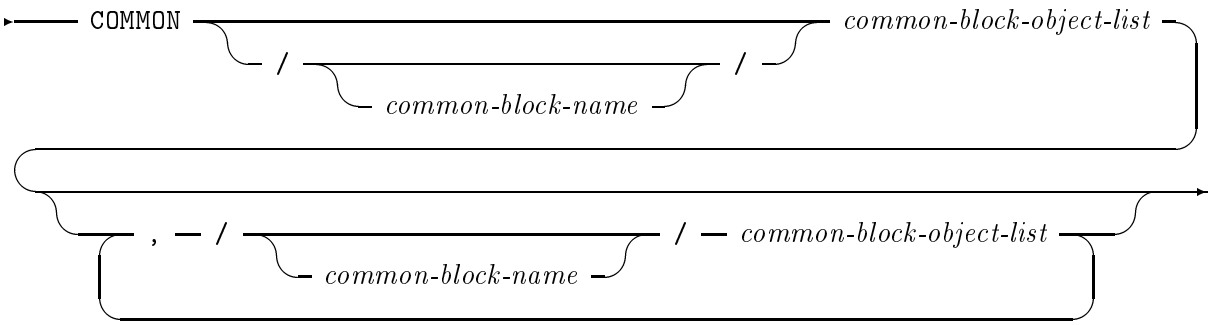
R547 *equivalence-set*



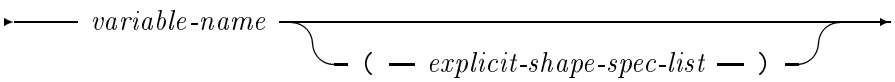
R548 *equivalence-object*



R549 *common-stmt*

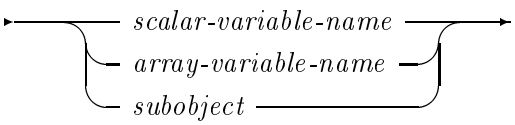


R550 *common-block-object*

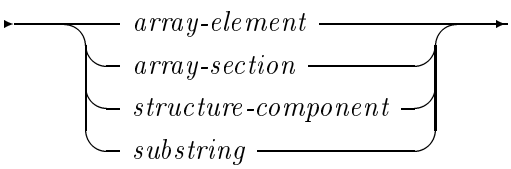


6 Verwendung von Datenobjekten

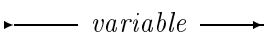
R601 *variable*



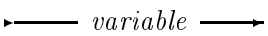
R602 *subobject*



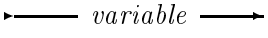
R603 *logical-variable*



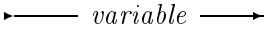
R604 *default-logical-variable*



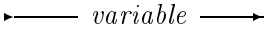
R605 *char-variable*



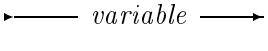
R606 *default-char-variable*



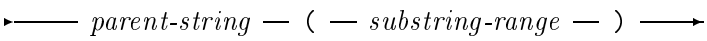
R607 *int-variable*



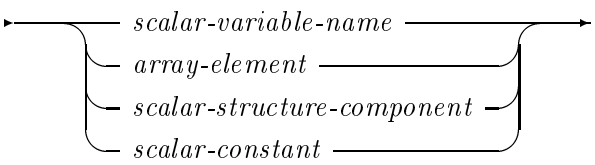
R608 *default-int-variable*



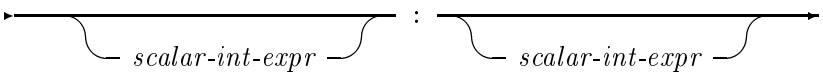
R609 *substring*



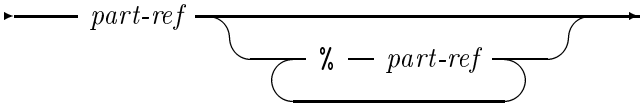
R610 *parent-string*



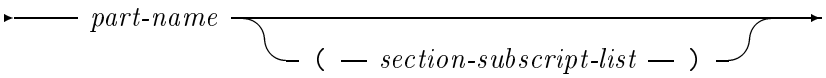
R611 *substring-range*



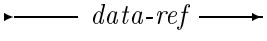
R612 *data-ref*



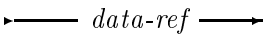
R613 *part-ref*



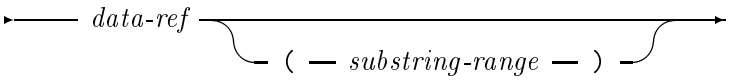
R614 *structure-component*



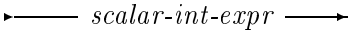
R615 *array-element*



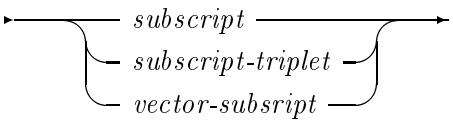
R616 *array-section*



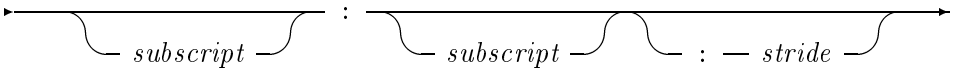
R617 *subscript*



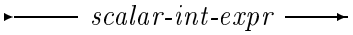
R618 *section-subscript*



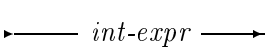
R619 *subscript-triplet*



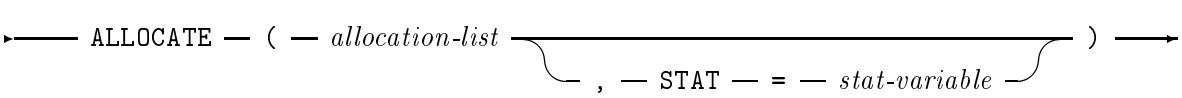
R620 *stride*



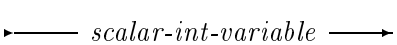
R621 *vector-subscript*



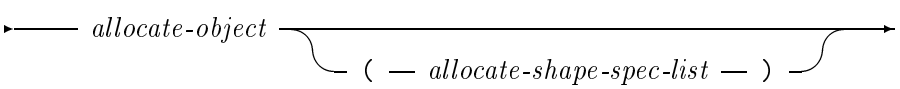
R622 *allocate-stmt*



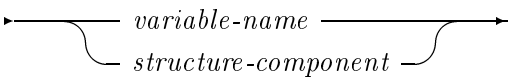
R623 *stat-variable*



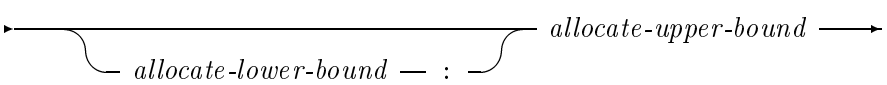
R624 *allocation*



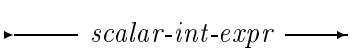
R625 *allocate-object*



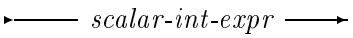
R626 *allocate-shape-spec*



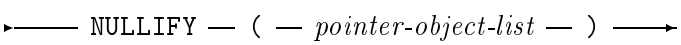
R627 *allocate-lower-bound*



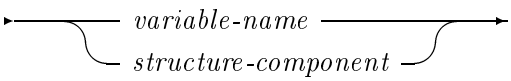
R628 *allocate-upper-bound*



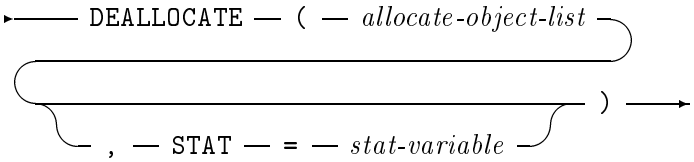
R629 *nullify-stmt*



R630 *pointer-object*

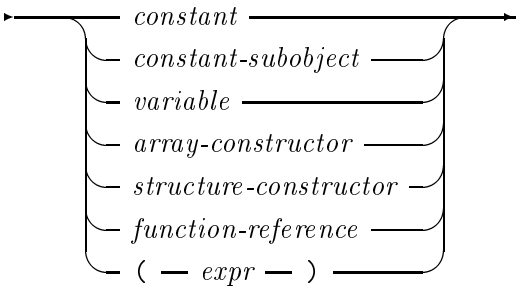


R631 *deallocate-stmt*

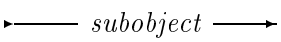


7 Ausdrücke und Zuweisungen

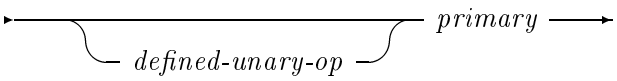
R701 *primary*



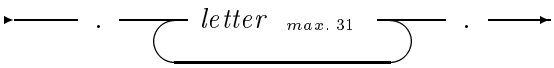
R702 *constant-subobject*



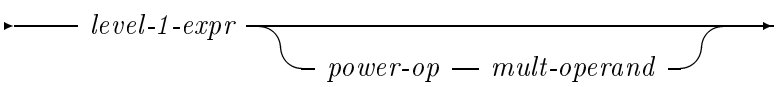
R703 *level-1-expr*



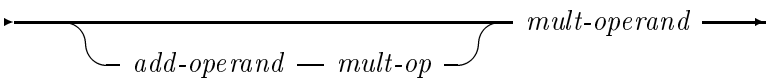
R704 *defined-unary-op*



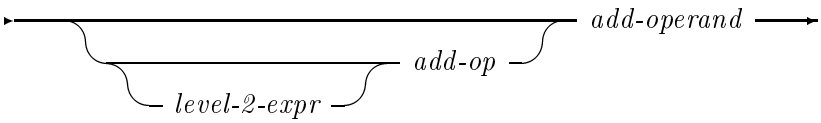
R705 *mult-operand*



R706 *add-operand*

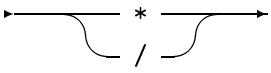


R707 *level-2-expr*

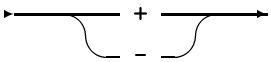


R708 *power-op*

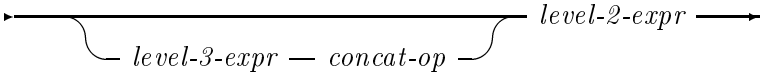
R709 *mult-op*



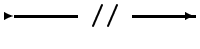
R710 *add-op*



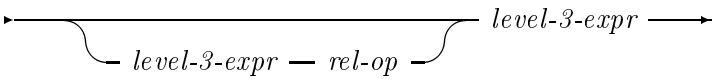
R711 *level-3-expr*



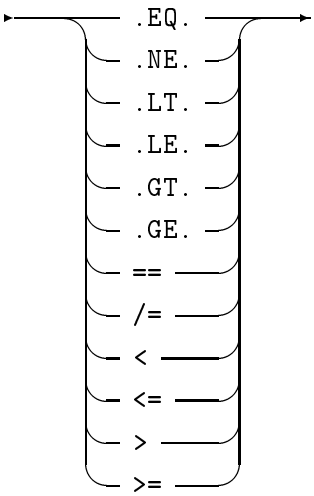
R712 *concat-op*



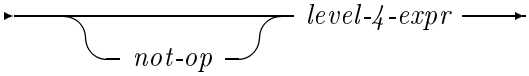
R713 *level-4-expr*



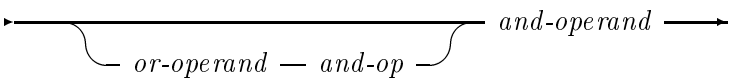
R714 *rel-op*



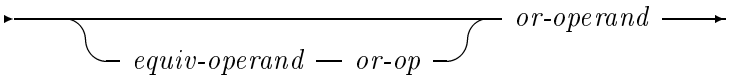
R715 *and-operand*



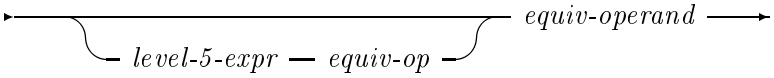
R716 *or-operand*



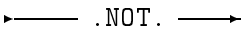
R717 *equiv-operand*



R718 *level-5-expr*

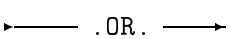


R719 *not-op*

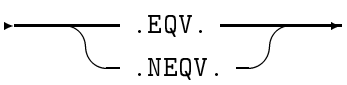


R720 *and-op*

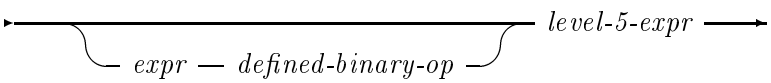
R721 *or-op*



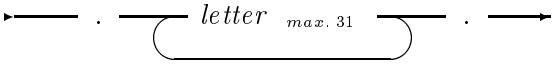
R722 *equiv-op*



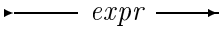
R723 *expr*



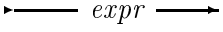
R724 *defined-binary-op*



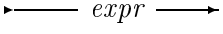
R725 *logical-expr*



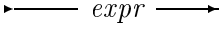
R726 *char-expr*



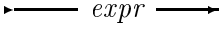
R727 *default-char-expr*



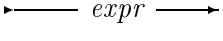
R728 *int-expr*



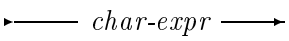
R729 *numeric-expr*



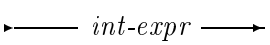
R730 *initialization-expr*



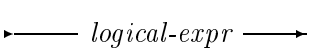
R731 *char-initialization-expr*



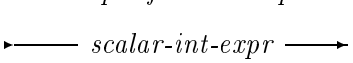
R732 *int-initialization-expr*



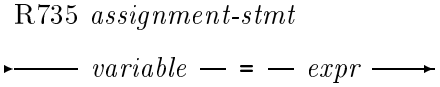
R733 *logical-initialization-expr*



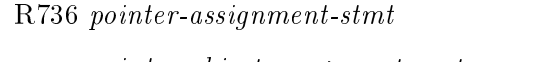
R734 *specification-expr*



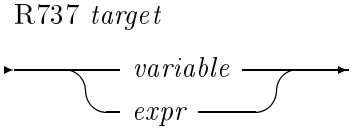
R735 *assignment-stmt*



R736 *pointer-assignment-stmt*

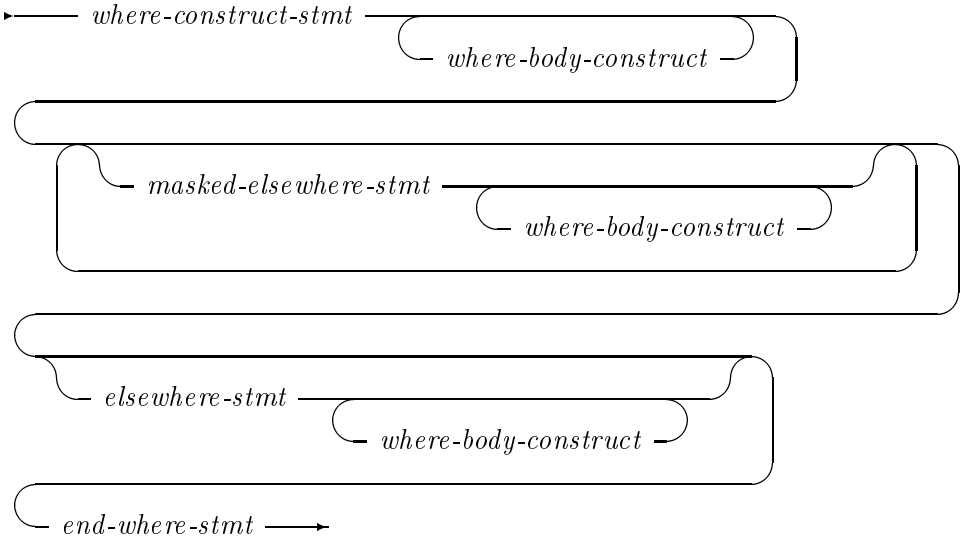


R737 *target*

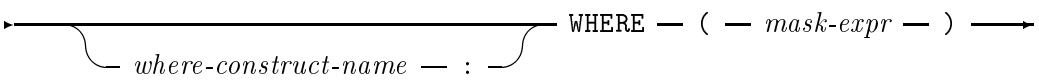


R738 *where-stmt*

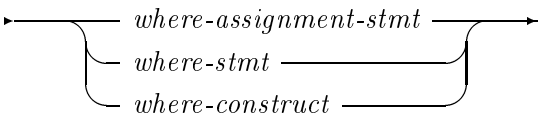
R739 *where-construct*



R740 *where-construct-stmt*

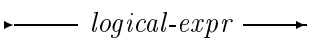


R741 *where-body-construct*

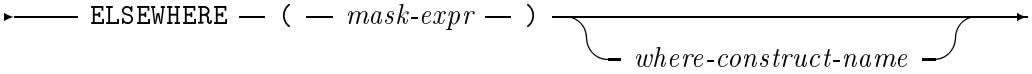


R742 *where-assignment-stmt*

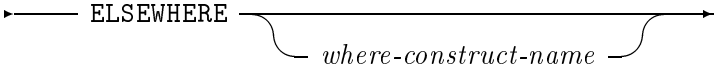
R743 *mask-expr*



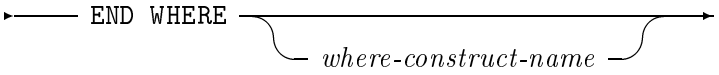
R744 *masked-elsewhere-stmt*



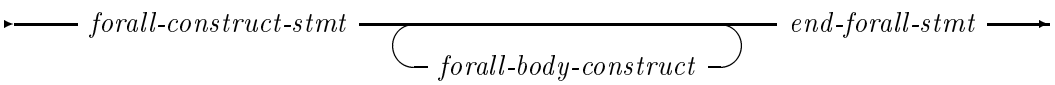
R745 *elsewhere-stmt*



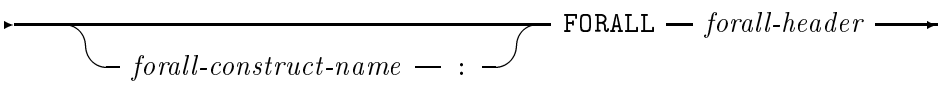
R746 *end-where-stmt*



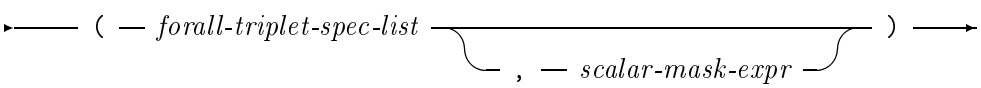
R747 *forall-construct*



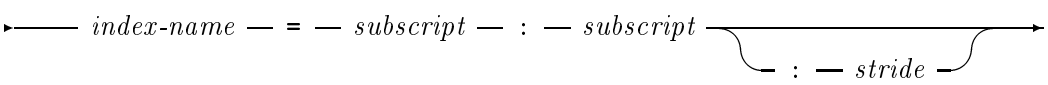
R748 *forall-construct-stmt*



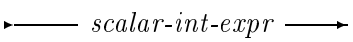
R749 *forall-header*



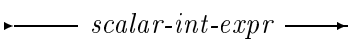
R750 *forall-triplet-spec*



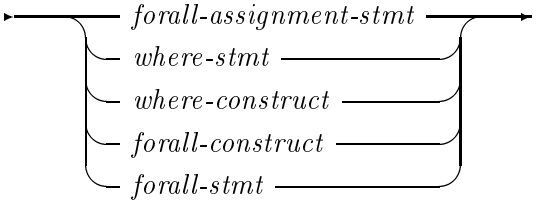
R617 *subscript*



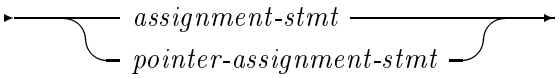
R603 *stride*



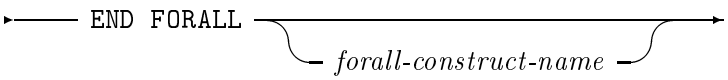
R751 *forall-body-construct*



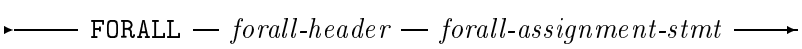
R752 *forall-assignment-stmt*



R753 *end-forall-stmt*

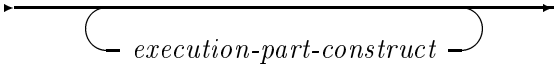


R754 *forall-stmt*

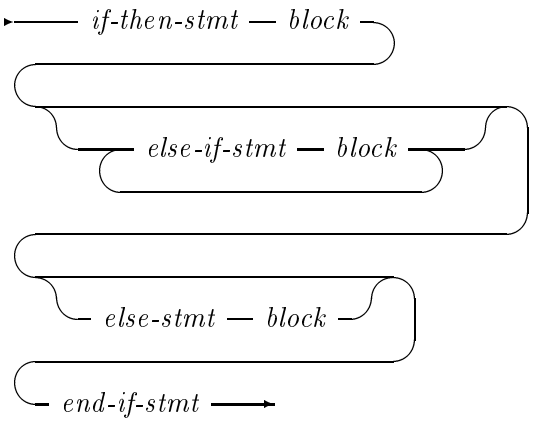


8 Ablaufsteuerung

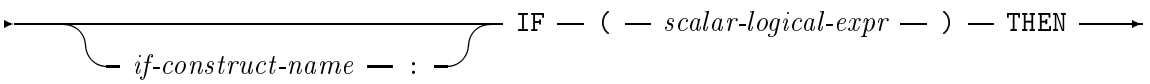
R801 *block*



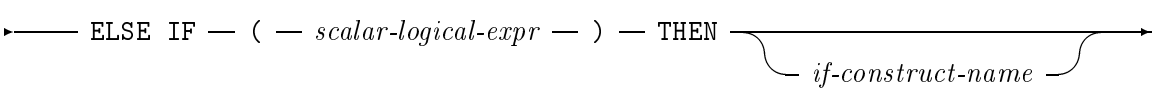
R802 *if-construct*



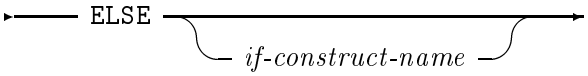
R803 *if-then-stmt*



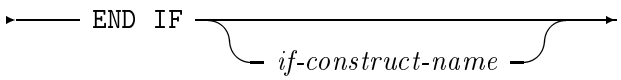
R804 *else-if-stmt*



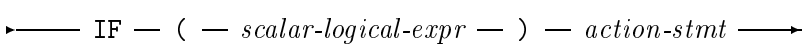
R805 *else-stmt*



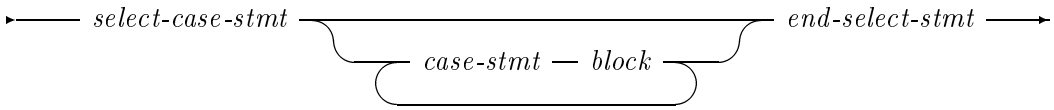
R806 *end-if-stmt*



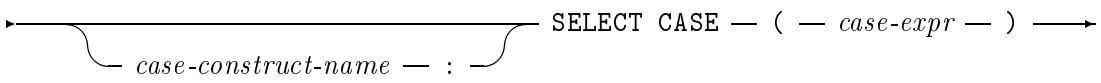
R807 *if-stmt*



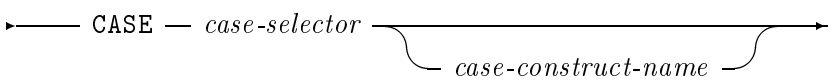
R808 *case-construct*



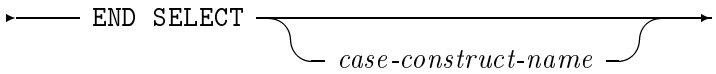
R809 *select-case-stmt*



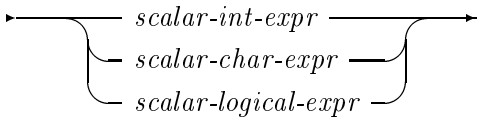
R810 *case-stmt*



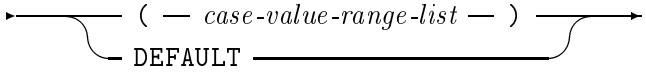
R811 *end-select-stmt*



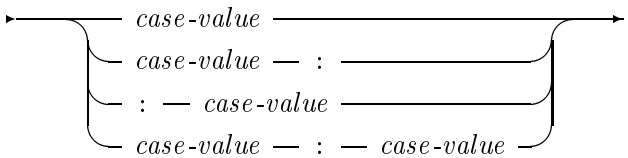
R812 *case-expr*



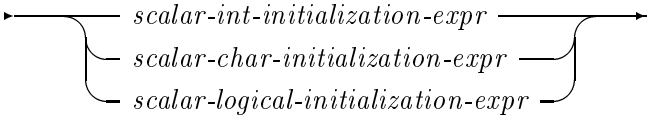
R813 *case-selector*



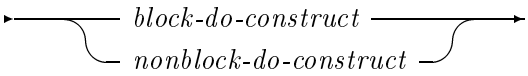
R814 *case-value-range*



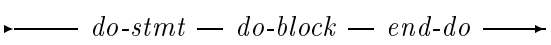
R815 *case-value*



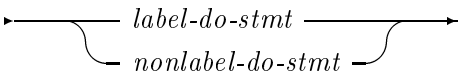
R816 *do-construct*



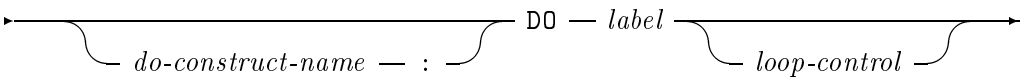
R817 *block-do-construct*



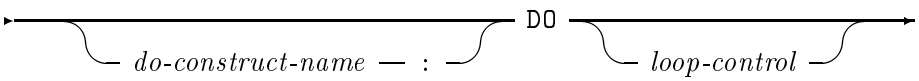
R818 *do-stmt*



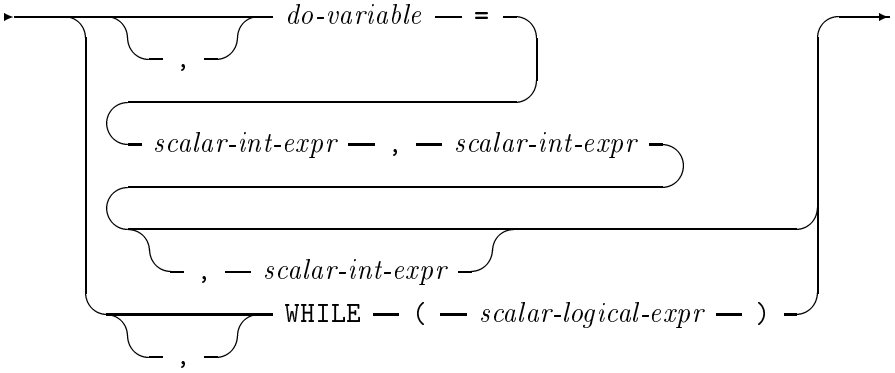
R819 *label-do-stmt*



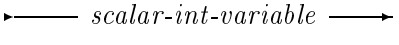
R820 *nonlabel-do-stmt*



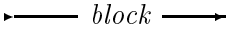
R821 *loop-control*



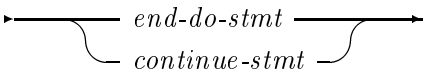
R822 *do-variable*



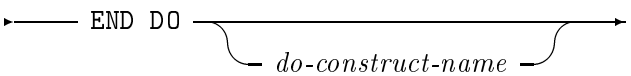
R823 *do-block*



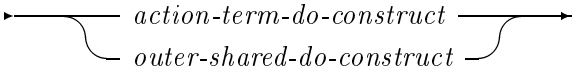
R824 *end-do*



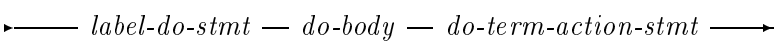
R825 *end-do-stmt*



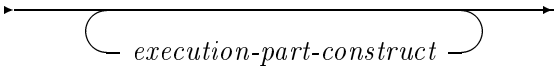
R826 *nonblock-do-construct*



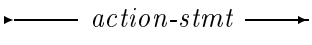
R827 *action-term-do-construct*



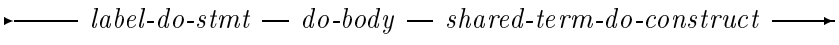
R828 *do-body*



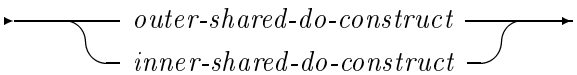
R829 *do-term-action-stmt*



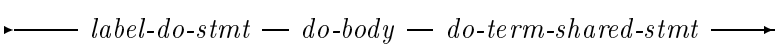
R830 *outer-shared-do-construct*



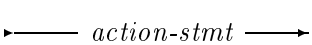
R831 *shared-term-do-construct*



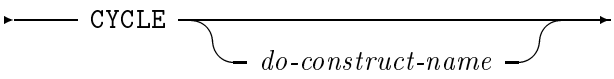
R832 *inner-shared-do-construct*



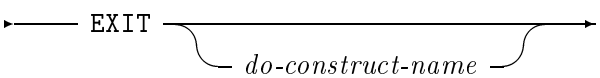
R833 *do-term-shared-stmt*



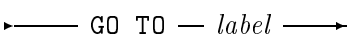
R834 *cycle-stmt*



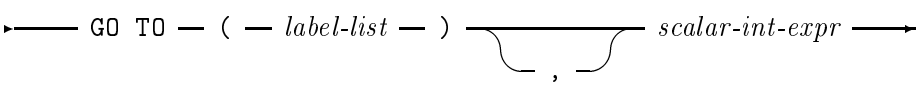
R835 *exit-stmt*



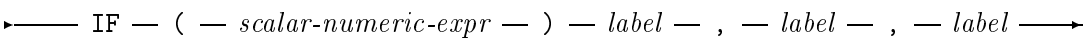
R836 *goto-stmt*



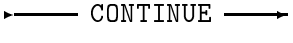
R837 *computed-goto-stmt*



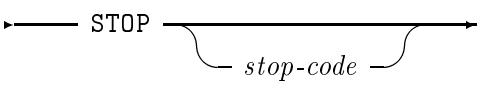
R838 *arithmetic-if-stmt*



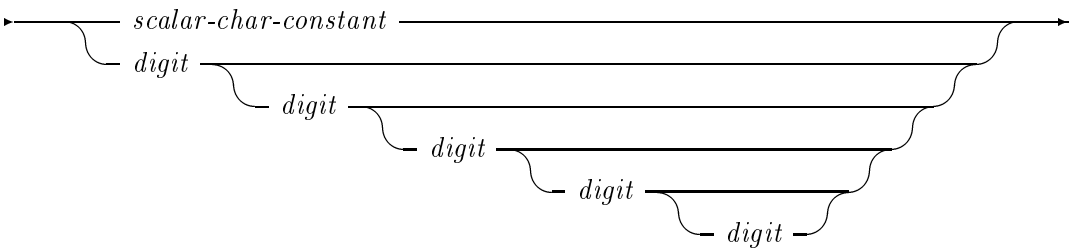
R839 *continue-stmt*



R840 *stop-stmt*

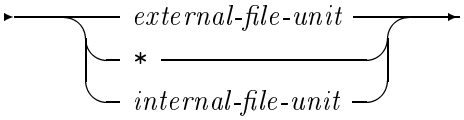


R841 *stop-code*

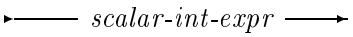


9 Ein-/Ausgabe-Anweisungen

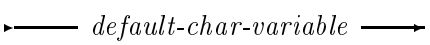
R901 *io-unit*



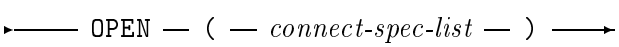
R902 *external-file-unit*



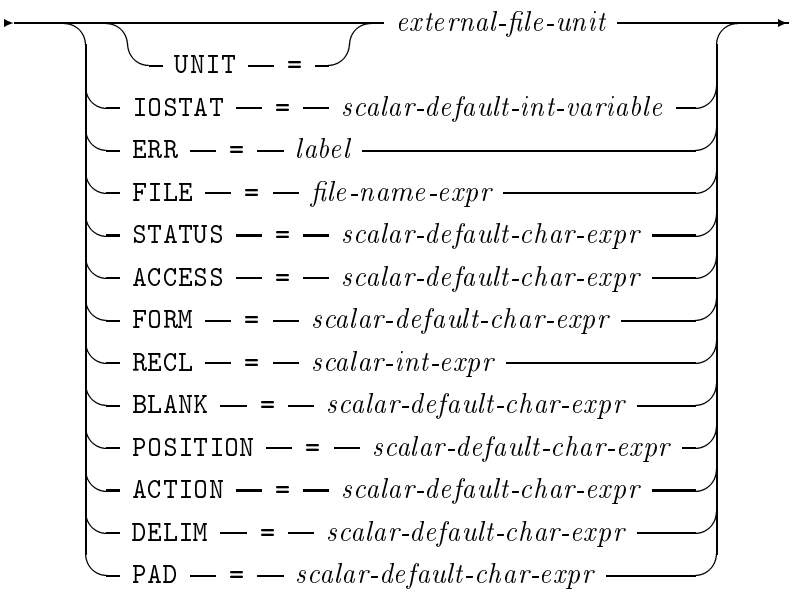
R903 *internal-file-unit*



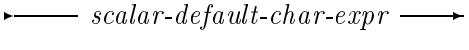
R904 *open-stmt*



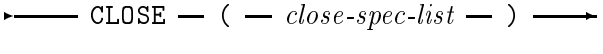
R905 *connect-spec*



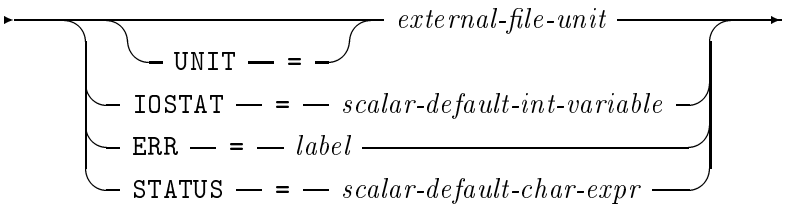
R906 *file-name-expr*



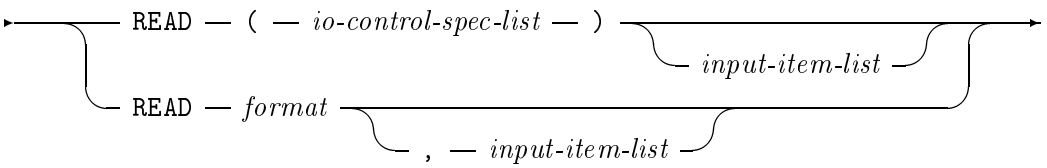
R907 *close-stmt*



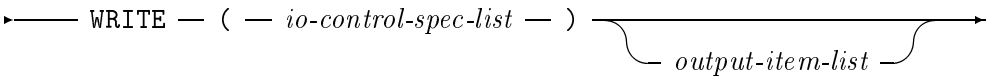
R908 *close-spec*



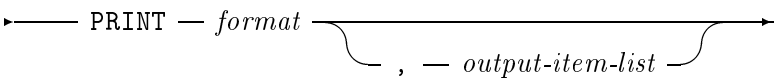
R909 *read-stmt*



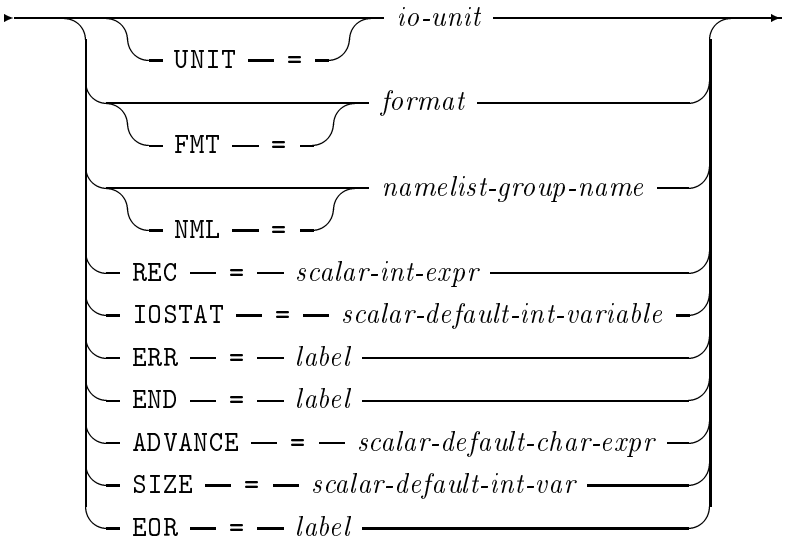
R910 *write-stmt*



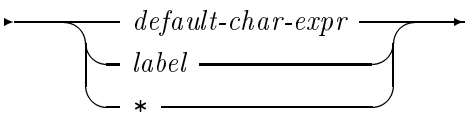
R911 *print-stmt*



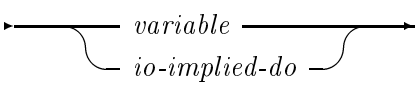
R912 *io-control-spec*



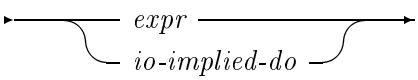
R913 *format*



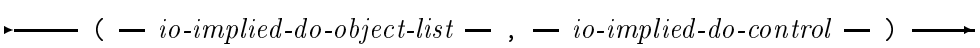
R914 *input-item*



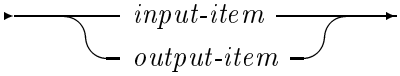
R915 *output-item*



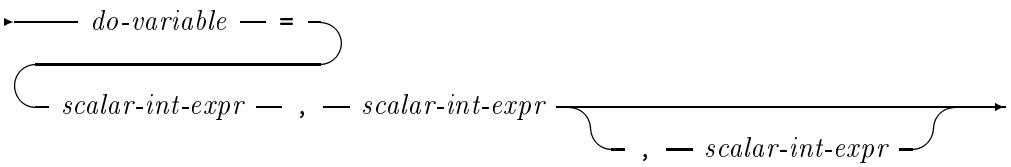
R916 *io-implied-do*



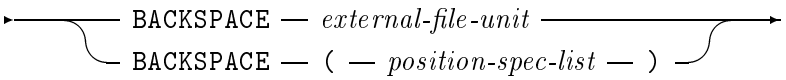
R917 *io-implied-do-object*



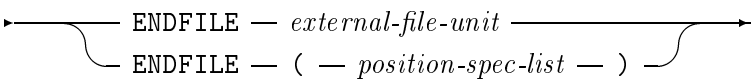
R918 *io-implied-do-control*



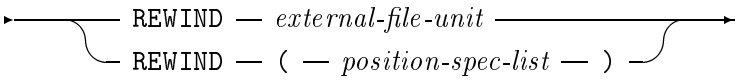
R919 *backspace-stmt*



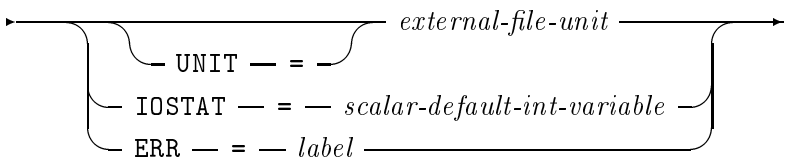
R920 *endfile-stmt*



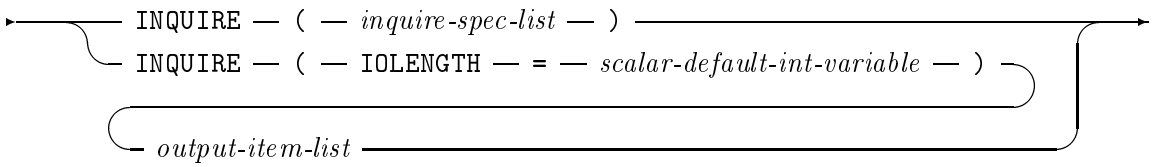
R921 *rewind-stmt*



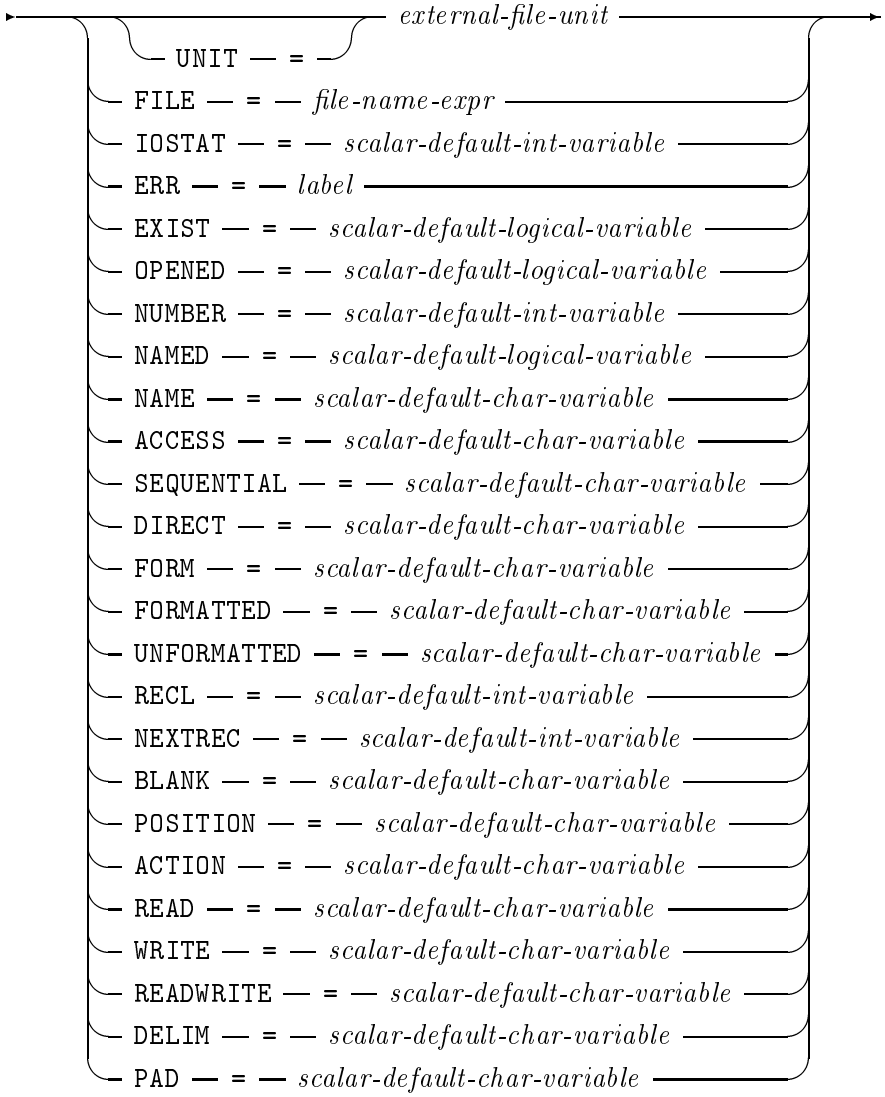
R922 *position-spec*



R923 *inquire-stmt*

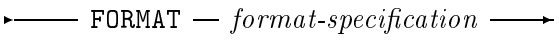


R924 *inquire-spec*

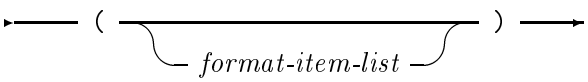


10 Ein-/Ausgabe-Aufbereitung

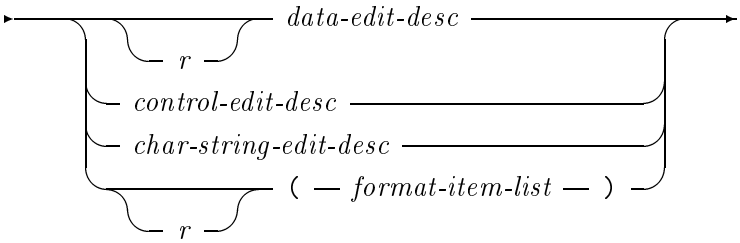
R1001 *format-stmt*



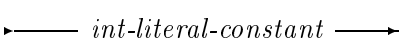
R1002 *format-specification*



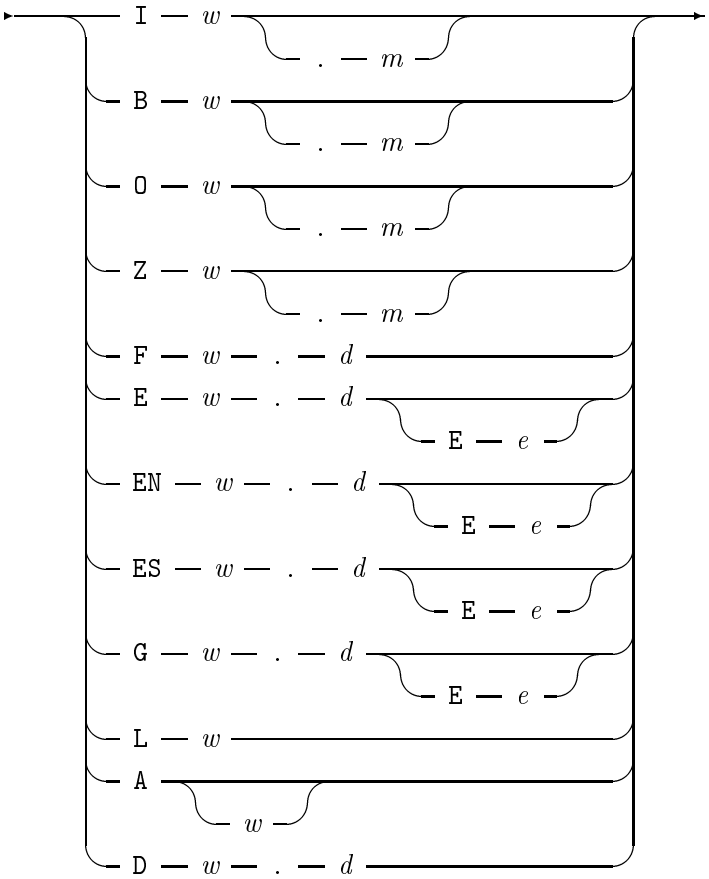
R1003 *format-item*



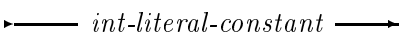
R1004 *r*



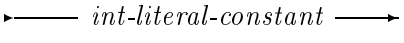
R1005 *data-edit-desc*



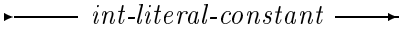
R1006 *w*



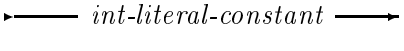
R1007 *m*



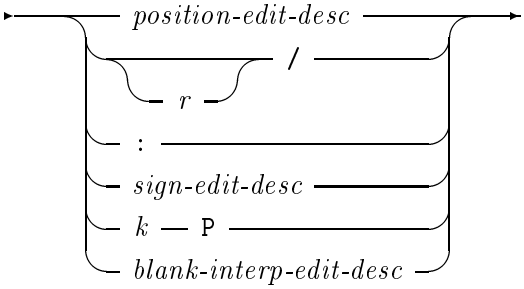
R1008 *d*



R1009 *e*

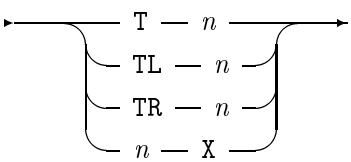


R1010 *control-edit-desc*

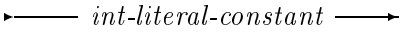


R1011 *k*

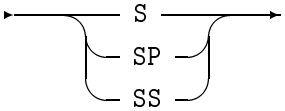
R1012 *position-edit-desc*



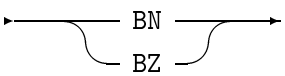
R1013 *n*



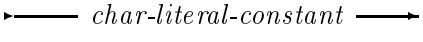
R1014 *sign-edit-desc*



R1015 *blank-interp-edit-desc*

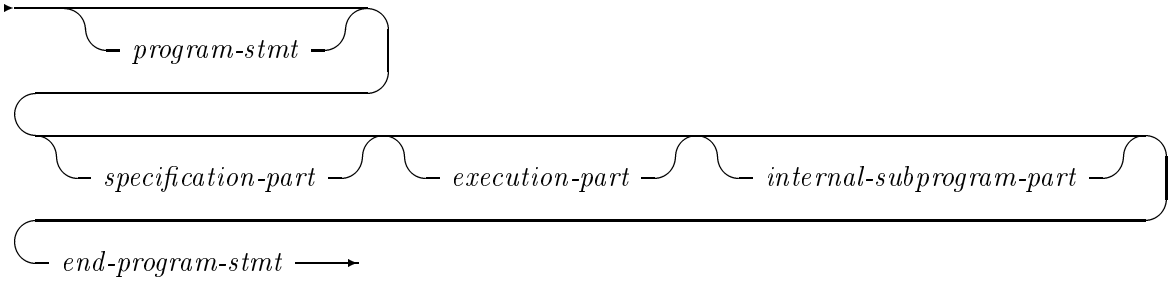


R1016 *char-string-edit-desc*

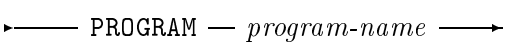


11 Programmeinheiten

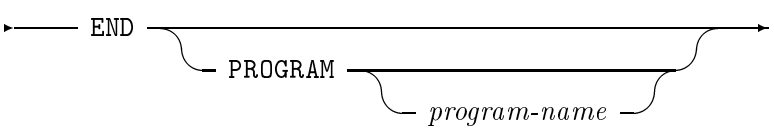
R1101 *main-program*



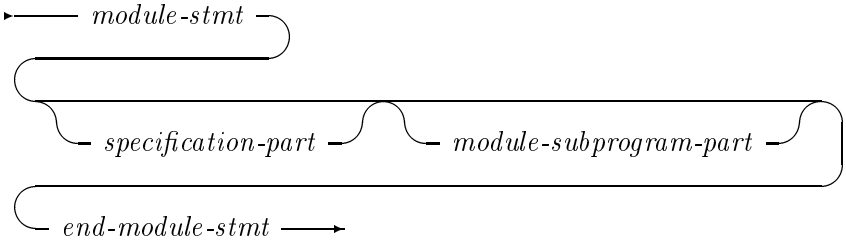
R1102 *program-stmt*



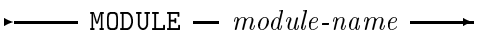
R1103 *end-program-stmt*



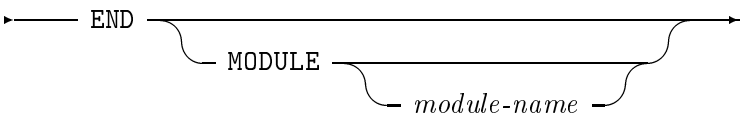
R1104 *module*



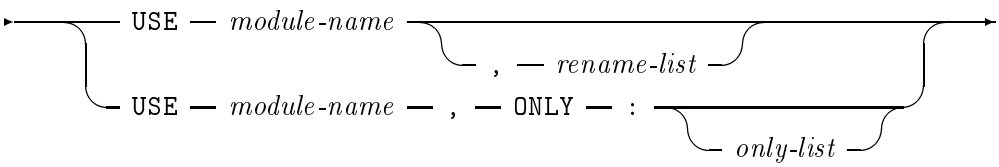
R1105 *module-stmt*



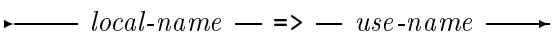
R1106 *end-module-stmt*



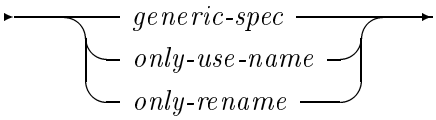
R1107 *use-stmt*



R1108 *rename*

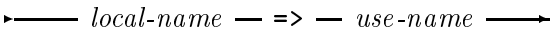


R1109 *only*

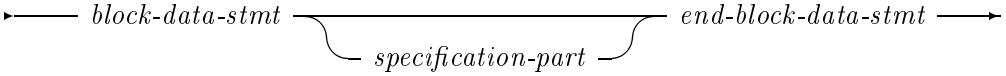


R1110 *only-use-name*

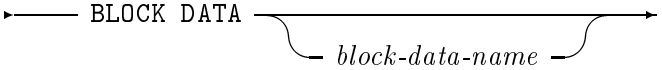
R1111 *only-rename*



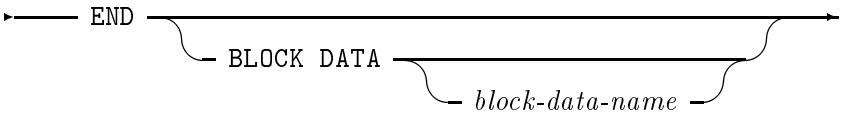
R1112 *block-data*



R1113 *block-data-stmt*

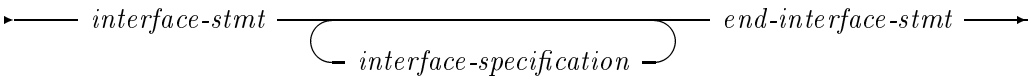


R1114 *end-block-data-stmt*

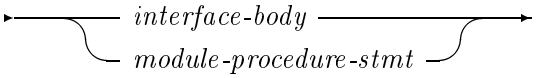


12 Prozeduren

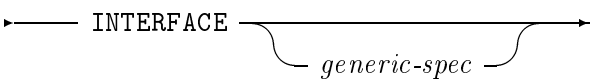
R1201 *interface-block*



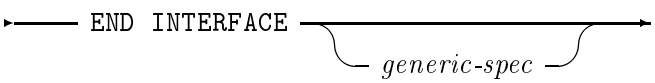
R1202 *interface-specification*



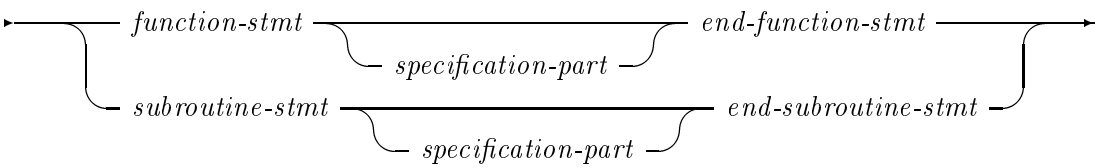
R1203 *interface-stmt*



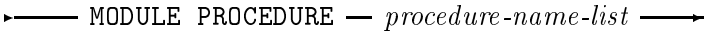
R1204 *end-interface-stmt*



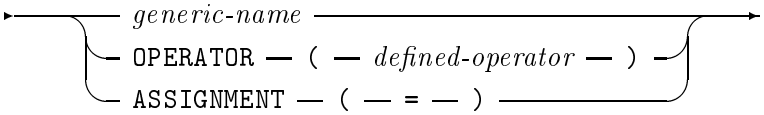
R1205 *interface-body*



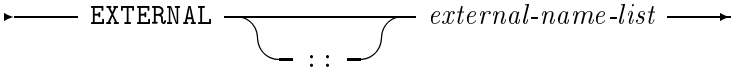
R1206 *module-procedure-stmt*



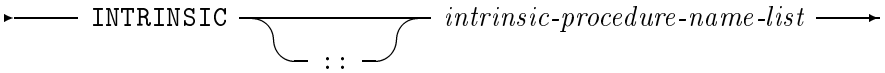
R1207 *generic-spec*



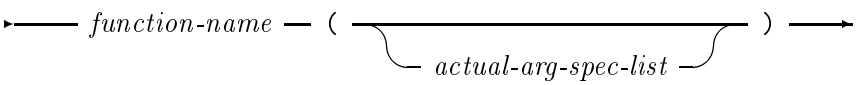
R1208 *external-stmt*



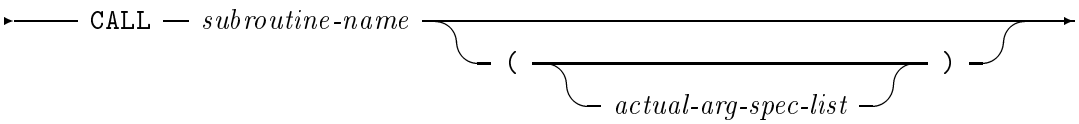
R1209 *intrinsic-stmt*



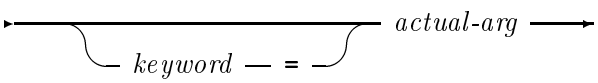
R1210 *function-reference*



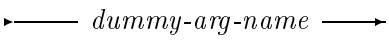
R1211 *call-stmt*



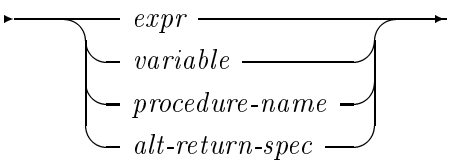
R1212 *actual-arg-spec*



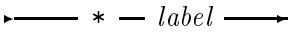
R1213 *keyword*



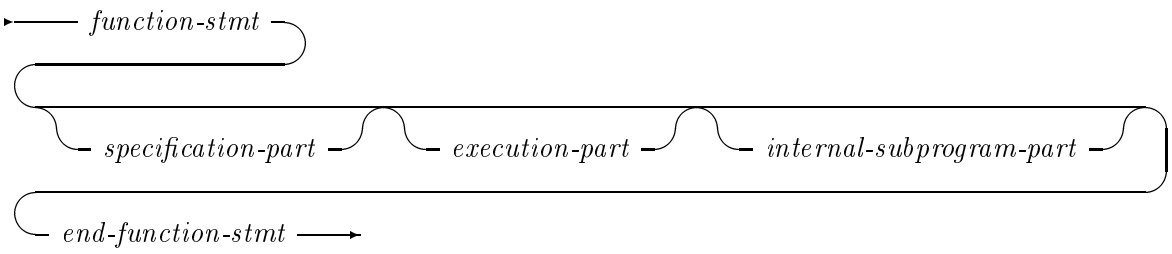
R1214 *actual-arg*



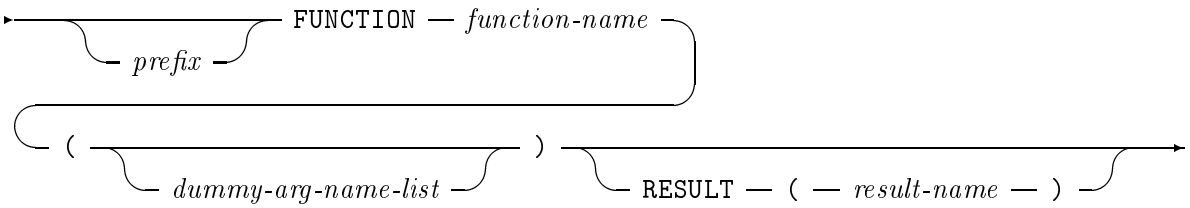
R1215 *alt-return-spec*



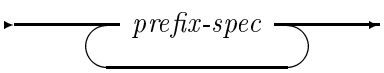
R1216 *function-subprogram*



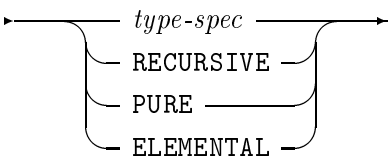
R1217 *function-stmt*



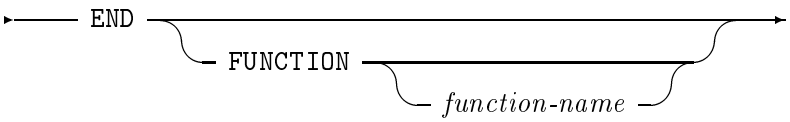
R1218 *prefix*



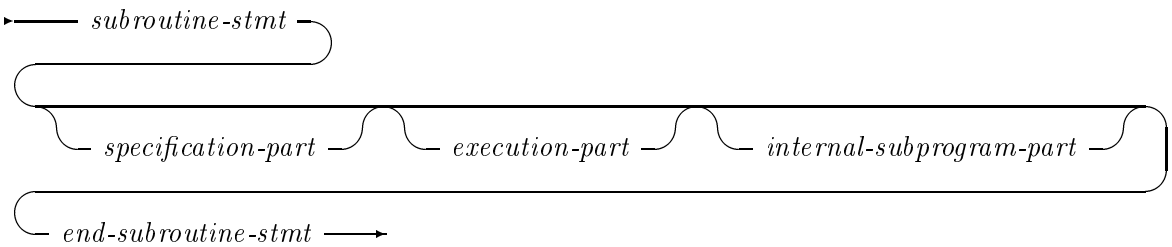
R1219 *prefix-spec*



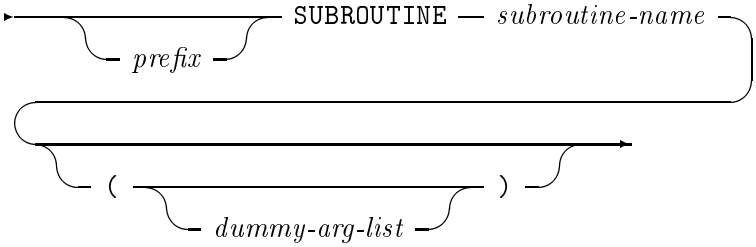
R1220 *end-function-stmt*



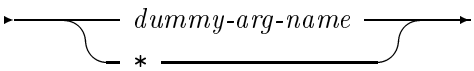
R1221 *subroutine-subprogram*



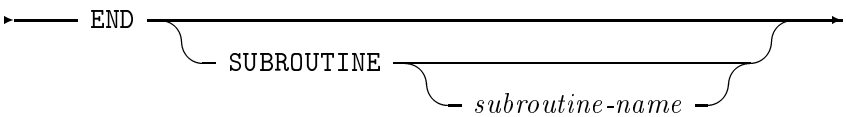
R1222 *subroutine-stmt*



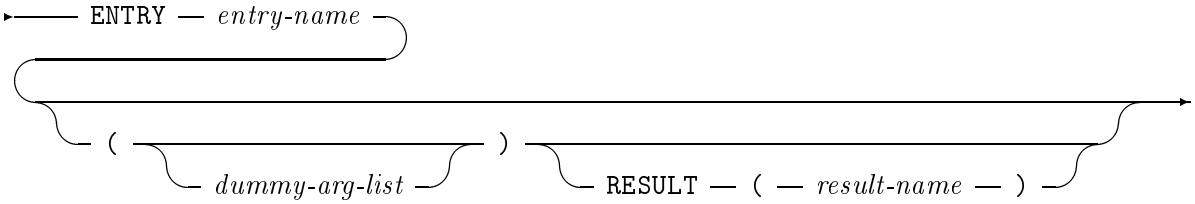
R1223 *dummy-arg*



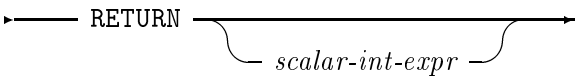
R1224 *end-subroutine-stmt*



R1225 *entry-stmt*



R1226 *return-stmt*



R1227 *contains-stmt*

R1228 *stmt-function-stmt*