SETL Newsletter # 173

Leonard Varek
April 30, 1976

SIMPLIFYING AND EXTENDING THE SETL TYPE CALCULUS

IN HIS ORIGINAL TYPE LATTICE TENENBAUM DRAWS A DISTINCTION BETWEEN ELEMENTARY TYPES (SUCH AS INTEGERS AND AND STRINGS) AND STRUCTURED TYPES (SETS AND TUPLES) WHEN HE REPRESENTS THEM IN TYPE DESCRIPTORS. THIS DISTINCTION BECOMES PARTICULARLY CUMBERSOME WHEN TREATING THE ELEMENTARY TYPES REPRESENTING THE NULL SET AND NULL TUPLE, SINCE THESE SHOULD BE TREATED AS SPECIAL CASES OF SET AND KNOWN-LENGTH TUPLE AND SHOULD APPEAR BELOW -SET- AND -KNT- IN A LATTICE OF ALL SETL TYPES.

ALSO A PROBLEM ARE THE MINIMUM AND MAXIMUM TYPES, TZ AND TG, WHICH OUGHT TO RESPECTIVELY EXCLUDE AND INCLUDE ALL OTHER TYPES - BOTH ELEMENTARY AND STRUCTURED.

IT IS POSSIBLE TO REDESIGN THE TYPE DESCRIPTORS AND TYPE LATTICE IN ORDER TO REMOVE THESE INCONSISTENCIES. THE RESULT IS A TYPE CALCULUS WHICH IS CONSIDERABLY SIMPLER BECAUSE STRUCTURED AND ELEMENTARY TYPES FIT INTO THE SAME LATTICE. FURTHERMORE, THIS LATTICE IS MADE AS NEARLY BOOLEAN AS POSSIBLE IN ORDER TO AID IN IMPLEMENTING THE LATTICE OPERATIONS -CON- AND -DIS-.

THE SUBLATTICE OF STRUCTURED TYPES

PREVIOUSLY, THE SUBLATTICE OF STRUCTURED TYPES CONSISTED OF ONLY THREE TYPES AND THE TWO IDENTITIES TZ AND TG. THIS LATTICE IS ILLUSTRATED IN FIGURE 1.

A MAJOR LIMITATION OF THIS REPRESENTATION IS THAT THE DISJUNCTION OF -SET- AND -UNT- IS -TG-. IN OTHER WORDS, IF WE CANNOT DISTINGUISH WHETHER A VARIABLE IS A SET OR A TUPLE WE MUST IGNORE THE FACT THAT IT IS CERTAINLY ONE OF THESE AND ASSUME THAT IT CAN BE ANY TYPE AT ALL.

THIS PROBLEM IS EASILY HANDLED BY CREATING A NEW POINT IN THE TYPE LATTICE WHICH LIES ABOVE SET AND UNT, BUT NOT ABOVE ANY OTHER TYPES. LET US CALL THIS NEW TYPE TG*. ALTHOUGH IT IS THE DISJUNCTION OF STRUCTURED TYPES, TG* WILL NOT ITSELF RETAIN ANY INFORMATION ABOUT THE THE COMPONENTS OF THE STRUCTURE; IT MERELY SERVES TO INDICATE THAT THE TYPE CAN BE EITHER SET OR TUPLE.

ONE APPLICATION OF TUPLES IS TO REPRESENT MAPS WHICH HAVE AS THEIR DOMAIN THE POSITIVE INTEGERS LESS THAN SOME RELATIVELY SMALL VALUE. THE SYNTAX FOR USING TUPLES IN THIS WAY DIFFERS VERY LITTLE FROM THE SYNTAX FOR USING SETS AS MAPS. THUS, IT MAY HAPPEN THAT THE TYPEFINDER CAN DETERMINE THAT A VALUE IS EITHER A MAP OR A TUPLE. IN THIS CASE, IT WOULD BE HELPFUL TO RETAIN SOME STRUCTURAL INFORMATION AS WELL AS THE FACT THAT THE TYPE IS EITHER SET OR TUPLE. FOR EXAMPLE, SUCH INFORMATION COULD INCLUDE THE TYPE OF THE OBJECTS IN THE DOMAIN AND RANGE (IMAGE) OF THE MAP. IN ORDER TO DO THIS IT IS NECESSARY TO HAVE A REPRESENTATION FOR THE DISJUNCTION OF TUPLES AND MAPS WHICH IS LESS GENERAL THAN TG*. HENCE, IT IS DESIREABLE TO DISTINGUISH BETWEEN SETS USED AS MAPS AND OTHER SETS, SO THAT THE DISJUNCTION OF TUPLES WITH MAPS CAN BE MADE DISTINCT FROM TG*.

ANOTHER MOTIVATION FOR HAVING TWO TYPES TO REPRESENT SETS COMES FROM THE PLANNED DATA STRUCTURES FOR IMPLEMENTING SETS. SINCE THERE WILL BE SEVERAL SUCH DATA STRUCTURES -

THE CHOICE DEPENDING ON HOW A SET IS USED - IT MAKES SENSE TO HELP IN THIS DECISION BY HAVING THE TYPEFINDER DETERMINE HOW A SET WILL BE USED.

THE PRECEDING TWO CONSIDERATIONS PROVIDE THE JUSTIFICATION FOR INTRODUCING THE FOLLOWING NEW TYPES INTO THE TYPE LATTICE:

MAP - A SET OF PAIRS WHICH IS USED AS A MAP.

SET - A SET NOT USED AS A MAP.

MAPSET - THE DISJUNCTION OF MAP AND SET. A SET POSSIBLY CONTAINING NON-PAIRS, BUT WHICH MAY BE USED AS A MAP.

MAPTUP - THE DISJUNCTION OF MAP AND EITHER UNT OR KNT.

FIGURE 2 SHOWS THE REVISED TYPE LATTICE. THE POINT -SETTUP- HAS BEEN ADDED IN ORDER TO AID IN THE IMPLEMENTATION OF THE TYPEFINDER ALGORITHM. THE CON AND DIS FUNCTIONS WILL BE MUCH SIMPLER IF THE LATTICE APPROXIMATES A BOOLEAN LATTICE, BECAUSE THE GROSS TYPES CAN THEN BE THOUGHT OF AS SETS OF BASIC TYPES AND BE MANIPULATED AS BIT STRINGS. WITH SETTUP ADDED, THE ONLY NON-BOOLEAN BEHAVIOUR EXHIBITED BY THE LATTICE OCCURS WHEN KNT IS DISJOINED WITH MAP, SET, OR MAPSET. THESE DISJUNCTIONS MUST BE CHECKED FOR AND FORCED TO BE MAPTUP, SETTUP AND TG*, RESPECTIVELY.

THE LATTICE POINTS TZ, TG*, AND SETTUP ARE ALL ACTUALLY ELEMENTARY TYPES - THAT IS, THEY HAVE NO ASSOCIATED INFORMATION ABOUT TYPES OF COMPONENTS - WHILE THE OTHER POINTS HAVE EXTRA STRUCTURAL INFORMATION, AS WILL BE EXPLAINED IN A LATER SECTION.

GROSS TYPES

IN THE DESCRIPTORS USED BY TENENBAUM FOR STRUCTURED TYPES THE GROSS TYPE AND ELEMENTARY ALTERNANDS ARE SEPARATE FIELDS IN THE DESCRIPTOR. HENCE, THE STRUCTURED TYPES HAD TO BE TREATED AS SPECIAL CASES NOT IN THE LATTICE OF ELEMENTARY TYPES. THE DISTINCTION BETWEEN GROSS TYPE AND ELEMENTARY ALTERNANDS HAS BEEN REMOVED, SO THAT THE -GROSSTYP- FIELD INCLUDES THE STRUCTURED AND ELEMENTARY ALTERNANDS.

A GROSS TYPE IS A FIELD IN A TYPE DESCRIPTOR, WHICH GIVES ALL TYPES THAT MIGHT BE ASSUMED BY A VALUE. IF ANY STRUCTURED TYPE IS INCLUDED IN THE GROSS TYPE, THE INFORMATION ABOUT THE TYPE AND NUMBER OF COMPONENTS WILL BE FOUND IN ONE OR TWO OTHER FIELDS OF THE TYPE DESCRIPTOR.

THE GROSS TYPE CAN BE THOUGHT OF AS BEING A SET OF BASIC TYPES. THE GENERATORS OF THE GROSS-TYPE LATTICE ARE (WITH A FEW EXCEPTIONS) SINGLETON SETS OF THESE BASIC TYPES, SO THAT DISJUNCTION OF GROSS TYPES IS CONCEPTUALLY SET UNION AND CONJUNCTION IS INTERSECTION. THE SINGLETON-SET GENERATORS ARE:

TU - UNDEFINED	TP - PROCEDURE
TA - BLANK ATOM	TL - LABEL
TSI - SHORT POSITIVE INTEGER	SET - SET NOT USED AS A MAP
TB1 - SINGLE BIT	MAP - SET OF PAIRS USED AS A MAP
TSC - SHORT CHARACTER STRING	KNT - TUPLE OF KNOWN LENGTH
TR - REAL NUMBER	

THE DOUBLETON-SET GENERATORS ARE:

TI - ANY INTEGER (INCLUDES TSI PROPERLY)

TSB - SHORT BIT STRING (INCLUDES TB1 PROPERLY)
TC - ANY CHARACTER STRING (INCLUDES TSC PROPERLY)
UNT - TUPLE OF UNKNOWN LENGTH (INCLUDES KNT PROPERLY)

AND THERE IS ONE GENERATOR WHICH HAS THREE ELEMENTS:

TB - ANY BIT STRING (PROPERLY INCLUDES TSB)

THE MOTIVATION FOR HAVING TWO AND THREE-ELEMENT GENERATORS IS THAT IT MAKES LITTLE SENSE TO TRY TO ISOLATE VALUES WHICH MUST BE LONG ITEMS. ALSO, IF WE LACK THE PRECISION TO KNOW WHICH OF TWO STRUCTURED TYPES A VALUE IS, WE CANNOT CLAIM IT TO BE A TUPLE OF KNOWN LENGTH; HENCE, KNT MUST LIE BELOW UNT IN THE LATTICE EVEN THOUGH WE CAN ATTACH NO MEANING TO UNT-KNT (SET DIFFERENCE).

THUS, ALL GROSS TYPES CAN BE REPRESENTED AS SUBSETS OF TG, THE SET OF ALL 16 BASIC TYPES. IT IS REASONABLE TO USE A 16-BIT STRING TO IMPLEMENT THIS REPRESENTATION.

IF IT IS LATER DECIDED NOT TO ATTEMPT TO DETECT SHORT STRINGS, TSB AND TSC CAN BE REMOVED FROM THE LATTICE - THEREBY REDUCING TC TO A SINGLETON AND TB TO A DOUBLETON. TB1 IS ALWAYS USEFUL BECAUSE THE LOGICAL CONNECTIVES (OR., AND., EXOR., ETC.) CAN ONLY OPERATE ON SINGLE BITS AND THE RELATIONAL OPERATORS CAN ONLY PRODUCE SINGLE BITS.

AUXILLIARY TYPE FIELDS

IN ADDITION TO THE -GROSSTYP- FIELD IN A TYPE DESCRIPTOR, A DESCRIPTOR WHICH HAS ONE OF THE SIX STRUCTURED TYPES IN ITS GROSS TYPE WILL HAVE ONE OR TWO FIELDS GIVING TYPE INFORMATION FOR THE COMPONENTS OF THE STUCTURED TYPE. WE NOW DESCRIBE THESE FIELDS.

WE WILL DENOTE BY T AN ARBITRARY TYPE DESCRIPTOR. -GROSSTYP(T)- IS THE FIELD WITHIN T WHICH CONTAINS THE GROSS TYPE. THE STRUCTURED PART OF THE GROSS TYPE IS -STRUCPART(GROSSTYP(T))- AND THIS IS JUST THE INTERSECTION OF THE GROSS TYPE WITH TG*. IN THE FOLLOWING TABLE TS DENOTES THE STRUCTURED PART OF THE GROSS TYPE OF T.

TS	AUXILLIARY FIELD	DESCRIPTION OF FIELD
SET,MAPSET	COMPTYP(T)	TYPE OF ELEMENTS OF THE SET
MAP,MAPTUP	COMPTYP(T)	TYPE OF VALUES IN IMAGE OF THE MAP
	DOMTYP(T)	TYPE OF VALUES IN DOMAIN OF THE MAP
UNT	COMPTYP(T)	TYPE OF COMPONENTS OF THE TUPLE
KNT	LENTYP(T)	LENGTH OF THE TUPLE
	COMPTYP(T)	TUPLE OF LENGTH LENTYP(T) CONTAINING THE TYPE OF EACH COMPONENT OF THE TUPLE

THE ROUTINES WHICH COMPUTE TYPE DISJUNCTIONS AND CONJUNCTIONS CAN USE BOOLEAN LATTICE OPERATIONS SUCH AS SET UNION AND INTERSECTION - WITH A FIXUP FOR THE NON-BOOLEAN BEHAVIOUR OF KNT - TO COMPUTE THE NEW GROSS TYPE. THEN, FOR STRUCTURED TYPES ONLY, OPERANDS CAN BE CONVERTED TO THE TYPE OF THE RESULT IF NECESSARY, AND THE AUXILLIARY FIELDS OF THE RESULTING TYPE DESCRIPTOR CAN BE EVALUATED.

FOR NOTATIONAL PURPOSES STRUCTURED TYPES WILL BE WRITTEN IN THE FOLLOWING WAY:

SET(COMPTYP),

MAPSET(COMPTYP),
UNT(COMPTYP),
MAP(COMPTYP,DOMTYP),
MAPTUP(COMPTYP,DOMTYP),
KNT(COMPTYP,LENTYP)

THE ALTERNATION (DISJUNCTION) OF A STRUCTURED TYPE (SAY A SET) WITH AN ELEMENTARY TYPE, T, CAN BE WRITTEN AS

T + SET(COMPTYP)

AND ITS TYPE DESCRIPTOR HAS AS ITS GROSSTYP THE UNION OF T AND SET. OF COURSE, THE COMPTYP FIELD IS THE SAME AS IF THE SET WERE NOT ALTERNATED WITH THE ELEMENTARY TYPE(S).

NULL TYPES

THE MOST NATURAL WAY TO ENSURE THAT THE NULL SET AND NULL TUPLE APPEAR AS SPECIAL CASES OF SETS AND TUPLES IN THE TYPE LATTICE IS TO REPRESENT THEM AS SUCH. THAT IS, RATHER THAN TREATING THE SPECIAL ELEMENTARY TYPES, TN AND TT, AS SPECIAL CASES, IT IS BETTER TO REMOVE TN AND TT FROM THE SET OF ELEMENTARY TYPES AND USE THE CONVENTION THAT THE COMPONENTS OF NULL STRUCTURES ARE UNDEFINED (I.E. HAVE THE GROSS TYPE -TU-).

HENCE, USING OUR PREVIOUSLY INTRODUCED NOTATION, THE NULL SET IS REPRESENTED BY

SET(TUNDEF)

WHERE -TUNDEF- IS THE TYPE DESCRIPTOR WITH A GROSS TYPE WHICH IS EXACTLY EQUAL TO TU. GENERALIZING THIS CONVENTION TO MAPS, WE GET THE FOLLOWING NULL TYPES:

MAPSET(TUNDEF)
MAP(TUNDEF,TUNDEF)
MAPTUP(TUNDEF,TUNDEF)

A TYPE WHICH IS KNOWN TO BE A NULL TUPLE COULD BE REPRESENTED BY -KNT(NULT,0)-. HOWEVER, IN ORDER TO FACILITATE CONVERTION OF NULL TUPLE FROM -KNT- TO -UNT- NULL TUPLES WILL BE REPRESENTED BY

KNT(<TUNDEF>,0) OR UNT(TUNDEF)

THE LATTER FORM IS PROVIDED IN ORDER TO ALLOW THE REPRESENTATION OF A TUPLE WHICH MAY EITHER BE NULL OR NON-NULL. SUCH A TUPLE COULD BE REPRESENTED BY

UNT(TUNDEF+T)

WHERE T IS THE TYPE OF ANY COMPONENT WHICH MAY BE PRESENT. SIMILARLY, A POSSIBLY NULL SET CAN BE REPRESENTED BY

SET(TUNDEF+T).

THE EXTENDED TYPE CALCULUS

FOR NON-STRUCTURED TYPES (I.E. ELEMENTARY TYPES) CONJUNCTION AND DISJUNCTION ARE COMPUTED SIMPLY BY TAKING THE INTERSECTION OR UNION OF THE GROSSTYP FIELDS OF THE OPERANDS. BIT STRING OPERATIONS CAN BE USED TO IMPLEMENT THIS EFFICIENTLY. WHEN STRUCTURED TYPES ARE COMBINED TO GIVE A STRUCTURED-TYPE RESULT, IT IS ALSO NECESSARY TO COMPUTE THE CONJUNCTION OR DISJUNCTION OF THE COMPONENTS OF THE OPERANDS. BEFORE THIS CAN BE DONE, HOWEVER, IT MAY BE NECESSARY TO CONVERT ONE OR BOTH OPERANDS TO THE TYPE OF THE

RESULT BEFORE THE COMPONENTS CAN BE OPERATED ON.

FOR AN EXAMPLE OF THIS TYPE CONVERSION, CONSIDER THE DISJUNCTION

MAP(T1,T2) DIS. UNT(T3)

THE GROSS TYPE OF THE RESULT IS -MAPTUP-, SO BOTH OPERANDS MUST BE CONVERTED TO THE RESULT TYPE. FOR MAP THIS CONVERSION IS TRIVIAL AND NEED NOT BE DONE EXPLICITLY BECAUSE MAP AND MAPTUP HAVE THE SAME STRUCTURE. HOWEVER, UNT(T3) MUST BE CONVERTED TO MAPTUP(T3,TSINT) - A MAP OF SHORT POSITIVE INTEGERS TO T3. HENCE, THE RESULT OF THE DISJUNCTION IS

MAPTUP(T1 DIS. T3, T2, DIS. TSINT)
= MAPTUP(T1↓T3, T2↓TSINT)

WHERE TSINT IS THE TYPE DESCRIPTOR HAVING A GROSS TYPE WHICH IS EXACTLY EQUAL TO TSI.

IF ONLY ONE OPERAND HAS ANY STRUCTURE INFORMATION, NO CONVERSION WILL EVER BE NECESSARY SINCE THE RESULT (IF IT IS A STRUCTURED TYPE) WILL HAVE THE SAME TYPE AS THE STRUCTURED OPERAND.

THE FOLLOWING PAGES GIVE THE SETL LANGUAGE SUBROUTINES FOR COMPUTING CONJUNCTIONS, DISJUNCTIONS, AND THE CONVERSION OF STRUCTURED TYPES.

```
$  LOW LEVEL FUNCTIONS FOR IMPLEMENTING THE TYPE CALCULUS

$  FOR THE DIS. AND CON. ROUTINES, THE ARGUMENTS
$  T1 AND T2 ARE TYPE DESCRIPTORS WITH THE FOLLOWING POSSIBLE
$  SUBFIELDS:
$     GROSSTYP  (CONTAINING ELEMPART AND STRUCPART)
$     COMPTYP   (FOR ALL STRUCTURED TYPES)
$     DOMTYP    (FOR MAPTUP AND MAP TYPES)
$     LENTYP    (FOR KNT TYPES)
$  THE MACRO (OR FUNCTION) MAKETYPE FORMS A TYPE DESCRIPTOR FROM
$  SPECIFIED SUBFIELDS.  ELEMENTARY AND STRUCTURED ARE PREDICATES ON
$  GROSSTYPES, WHICH INDICATE THE PRESENCE OR ABSENCE OF STRUCTURED
$  ALTERNANDS IN THE GROSSTYPES.
```

```
DEFINEF T1 DIS. T2;

 G = GROSSTYP(T1) + GROSSTYP(T2);
 IF (G*KNT EQ. KNT) AND. (G*MAPSET NE. TZ) THEN G = G + UNT;;
 IF ELEMENTARY(G) THEN RETURN MAKETYPE(G, OM., OM.);;

 $  RESULT IS STRUCTURED - CHECK TO SEE IF ONE DISJUNCT IS ELEMENTARY
 IF ELEMENTARY(GROSSTYP(T1)) THEN GROSSTYP(T2) = G; RETURN T2;;
 IF ELEMENTARY(GROSSTYP(T2)) THEN GROSSTYP(T1) = G; RETURN T1;;
 $  IF WE GOT THIS FAR, BOTH DISJUNCTS ARE STRUCTURED, SO IT WILL BE
 $  NECESSARY TO CALL DIS RECURSIVELY TO FIND TYPES OF COMPONENTS.
 $  ALSO, CONVERSION OF COMPONENT TYPE-STRUCTURE MAY BE NEEDED.
 C = STRUCPART(G);
 C1 = STRUCPART(GROSSTYP(T1));   C2 = STRUCPART(GROSSTYP(T2));
 IF (C EQ. MAPTUP) OR. (C EQ. MAP) THEN
    $ IF C EQ. MAPTUP, ONE OF T1 AND T2 MAY NEED TO BE CONVERTED
    $ FROM EITHER KNT OR UNT TO MAPTUP
    IF (C1 EQ. KNT) OR. (C1 EQ. UNT) THEN
       T1 = CONVERT(T1,MAPTUP);
    ELSEIF (C2 EQ. KNT) OR. (C2 EQ. UNT) THEN
       T2 = CONVERT(T2,MAPTUP);
    END IF;
    RETURN MAKETYPE(G, COMPTYP(T1) DIS. COMPTYP(T2),
                       DOMTYP(T1) DIS. DOMTYP(T2));
 END IF;

 IF (C EQ. MAPSET) OR. (C EQ. SET) OR. (C EQ. UNT) THEN
    $ IF C EQ. MAPSET, ONE OF T1 AND T2 MAY NEED TO BE CONVERTED FROM
    $ MAP FORMAT TO MAPSET FORMAT
    IF C1 EQ. MAP THEN T1 = CONVERT(T1,MAPSET);
    ELSEIF C2 EQ. MAP THEN T2 = CONVERT(T2,MAPSET); END IF;
    $ IF C EQ. UNT, ONE OF T1 AND T2 MAY NEED TO BE CONVERTED FROM
    $ KNT FORMAT TO UNT FORMAT
    IF C1 EQ. KNT THEN T1 = CONVERT(T1,UNT);
    ELSEIF C2 EQ. KNT THEN T2 = CONVERT(T1,UNT); END IF;
    RETURN MAKETYPE(G, COMPTYP(T1) DIS. COMPTYP(T2), OM.);
 END IF;

 $ C IS OF TYPE KNT IF WE GET THIS FAR
 IF LENTYP(T1) EQ. LENTYP(T2) THEN
    CT = <TU>;
    CT1 = COMPTYP(T1);      CT2 = COMPTYP(T2);
    (∨ 1 <= I <= LENTYP(T1)) CT(I) = CT1(I) DIS. CT2(I);;
    RETURN MAKETYPE(G, CT, LENTYP(T1));
 END IF;

 $ KNOWN-LENGTH TUPLES ARE OF DIFFERENT LENGTHS, SO BOTH MUST
 $ BE CONVERTED TO UNT
 T1 = CONVERT(T1,UNT);      T2 = CONVERT(T2,UNT);
 RETURN MAKETYPE(G+UNT, COMPTYP(T1) DIS. COMPTYP(T2), OM.);
END DIS;
```

```
DEFINEF T1 CON. T2;

 G = GROSSTYP(T1) * GROSSTYP(T2);
 IF ELEMENTARY(G) THEN RETURN MAKETYPE(G, OM., OM.);;

 $  RESULT IS STRUCTURED - CHECK TO SEE IF ONE CONJUNCT IS ELEMENTARY
 IF ELEMENTARY(GROSSTYP(T1)) THEN GROSSTYP(T2) = G; RETURN T2;;
 IF ELEMENTARY(GROSSTYP(T2)) THEN GROSSTYP(T1) = G; RETURN T1;;
 $  IF WE GOT THIS FAR, BOTH CONJUNCTS ARE STRUCTURED, SO IT WILL BE
 $  NECESSARY TO CALL CON RECURSIVELY TO FIND TYPES OF COMPONENTS.
 $  ALSO, CONVERSION OF COMPONENT TYPE-STRUCTURE MAY BE NEEDED.
 C = STRUCPART(G);
 C1 = STRUCPART(GROSSTYP(T1));
 C2 = STRUCPART(GROSSTYP(T2));
 IF (C EQ. MAPTUP) OR. (C EQ. MAP) THEN
    $ IF C EQ. MAP, ONE OF T1 AND T2 MAY NEED TO BE CONVERTED
    $ FROM MAPSET TO MAP
    IF C1 EQ. MAPSET THEN T1 = CONVERT(T1,MAP);
    ELSEIF C2 EQ. MAPSET THEN T2 = CONVERT(T2,MAP); END IF;
    RETURN MAKETYPE(G, COMPTYP(T1) CON. COMPTYP(T2),
                       DOMTYP(T1) CON. DOMTYP(T2));
 END IF;

 IF (C EQ. MAPSET) OR. (C EQ. SET) OR. (C EQ. UNT) THEN
    $ IF C EQ. SET, ONE OF T1 AND T2 MAY NEED TO BE CONVERTED FROM
    $ MAPSET FORMAT TO SET FORMAT
    IF C1 EQ. MAPSET THEN T1 = CONVERT(T1,SET);
    ELSEIF C2 EQ. MAPSET THEN T2 = CONVERT(T2,SET); END IF;
    $ IF C EQ. UNT, ONE OF T1 AND T2 MAY NEED TO BE CONVERTED FROM
    $ MAPTUP FORMAT TO UNT FORMAT
    IF C1 EQ. MAPTUP THEN T1 = CONVERT(T1,UNT);
    ELSEIF C2 EQ. MAPTUP THEN T2 = CONVERT(T1,UNT); END IF;
    RETURN MAKETYPE(G, COMPTYP(T1) CON. COMPTYP(T2), OM.);
 END IF;

 $ C IS OF TYPE KNT IF WE GET THIS FAR
 IF C1 NE. KNT THEN T1 = CONVERT(T1,LENTYP(T2));
 ELSEIF C2 NE. KNT THEN T2 = CONVERT(T2,LENTYP(T1));
 END IF;
 IF LENTYP(T1) EQ. LENTYP(T2) THEN
    CT = <TU>;
    CT1 = COMPTYP(T1);        CT2 = COMPTYP(T2);
    (V 1 <= I <= LENTYP(T1)) CT(I) = CT1(I) CON. CT2(I);;
    RETURN MAKETYPE(G, CT, LENTYP(T1));
 END IF;

 $ KNOWN-LENGTH TUPLES ARE OF DIFFERENT LENGTHS, SO THEIR
 $ CONJUNCTION MUST BE THE RELATIVE ZERO ELEMENT, TZSTRUC
 RETURN MAKETYPE(G * TGSTRUC, OM., OM.);
END CON;
```

```
DEFINEF CONVERT(T, NEWTYP);
$  CONVERTS STRUCTURED TYPE T TO NEW GROSS TYPE NEWTYP, WHERE
$  IF THE NEW GROSSTYPE SHOULD BE KNT, THE TUPLE LENGTH IS
$  PASSED VIA NEWTYP RATHER THAN THE VALUE -KNT-.
$  IT IS ASSUMED THAT ONLY VALID CONVERSIONS ARE BEING ATTEMPTED
  C = STRUCPART(GROSSTYP(T));

  IF C EQ. KNT THEN
     IF NEWTYP NE. UNT THEN RETURN T;;
     T = MAKETYPE(GROSSTYP(T)+UNT,
                  [DIS: 1<=I<=MAX(1,LENTYP(T))] COMPTYP(T)(I), OM.);
     IF NEWTYP EQ. UNT THEN RETURN T;
                       ELSE C = UNT;;
  END IF C;

  IF C EQ. UNT AND. NEWTYP EQ. MAPTUP THEN
    RETURN MAKETYPE(GROSSTYP(T)+MAPTUP, COMPTYP(T), TSI);
  END IF C;

  IF C EQ. UNT THEN
     IF NEWTYP EQ. UNT THEN RETURN T;;
     N = MAX(1, NEWTYP);
     RETURN MAKETYPE(GROSSTYP(T)*KNT, REPL(<COMPTYP(T)>,N), NEWTYP);
  END IF C;

  IF C EQ. MAP THEN
     IF NEWTYP IN. ≤MAP, MAPTUP≥ THEN RETURN T;;
     RETURN MAKETYPE(GROSSTYP(T)+MAPSET,
                     MAKETYPE(KNT,<DOMTYP(T),COMPTYP(T)>,2), OM.);
  END IF C;

  IF C EQ. MAPTUP THEN
     IF NEWTYP IN. ≤MAP, MAPTUP≥ THEN RETURN T;;
     T = MAKETYPE(GROSSTYP(T)*UNT, COMPTYP(T), OM.);
     IF NEWTYP EQ. UNT THEN RETURN T;
                       ELSE C = UNT;;
  END IF C;

  IF C EQ. MAPSET THEN
     IF NEWTYP IN. ≤SET, MAPSET≥ THEN RETURN T;;
     COMPONENT = COMPTYP(T);
     NEWCOMP = COMPONENT CON. MAKETYPE(KNT,<TGEN,TGEN>,2);
     IF COMPONENT CON. TUNDEF EQ. TUNDEF THEN
     $ ALLOW NULL SET TO BE CONVERTED TO NULL MAP
        NEWCOMP = NEWCOMP DIS. MAKETYPE(KNT,<TUNDEF,TUNDEF>,2);;
     IF GROSSTYP(NEWCOMP) EQ. TZ THEN RETURN NEWCOMP;;
     <C1, C2> = COMPTYP(NEWCOMP);
     RETURN MAKETYPE(GROSSTYP(T)*MAP, C2, C1);
  END IF C;

END CONVERT;
```

SUBLATTICES OF STRUCTURED TYPES

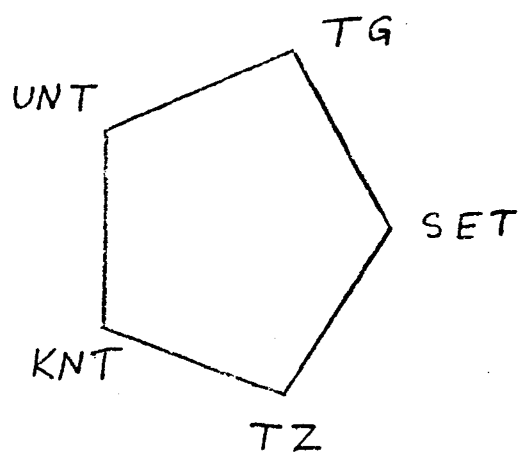

FIGURE 1 - TENENBAUM'S SUBLATTICE

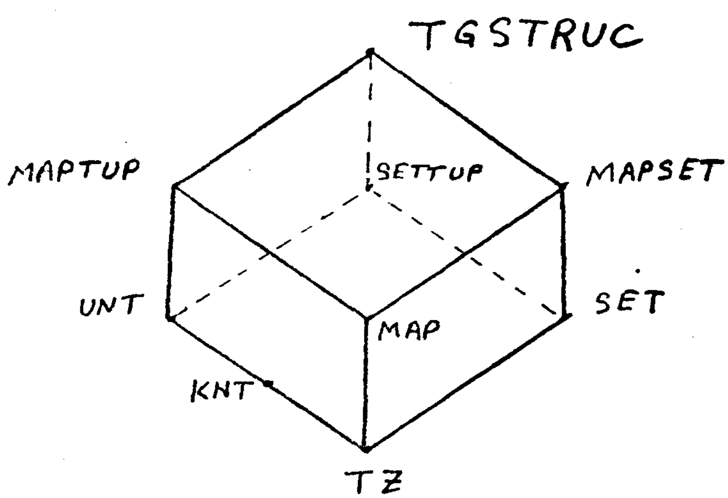

FIGURE 2 - REVISED SUBLATTICE