BAB III

KONVERSI FILE STEP-NC KE G CODE

3.1 PEMETAAN (*MAPPING*)

Langkah awal untuk melakukan proses konversi *file* STEP-NC ke G Code adalah dengan proses *mapping*. Proses *mapping* adalah proses memetakan hubungan antara standar ISO 6983 (G-Code) degan ISO 14649 (STEP). Dalam format ISO 14649 dibagi dalam beberapa grup yakni :

1. *Workpiece definition*
2. *Manufacturing features*
3. *Operations*
4. *Project*
5. *Function/Technology*
6. *Strategies*
7. *Placement/Length*
8. *Tools*

```
ISO-10303-21;

HEADER;
FILE_DESCRIPTION(('EXAMPLE OF NC PROGRAMME FOR TURNING: COMPLEX DESIGN.'),'1');
FILE_NAME('EXAMPLE1.STP',$,('ISO14649'),(''),'SUH','POSTECH', 'KOREA');
FILE_SCHEMA(('MACHINING_SCHEMA','TURNING_SCHEMA'));
ENDSEC;

DATA;
(* ***** Workpiece definition ***** *)
#1=WORKPIECE('SIMPLE WORKPIECE',#2,0.01,$,$,$,());
#2=MATERIAL('ST-50','STEEL',(#3));

(* ***** Manufacturing features ***** *)
#12=GENERAL_REVOLUTION('GENERAL_REVOLUTION_1',#1,(#20,#21),#194,#198,21.0,#200);
#13=ROUND_HOLE('HOLE1 FLAT BOTTOM',#1,(#26,#27,#28),#207,#215,#216,$,#217);

(* ***** Turning operations ***** *)
#20=CONTOURING_ROUGH($,$,'ROUGH GENERAL REVOLUTION1',30.000,$,#280,#61,#60,#130,#130,#131,0.5);
#21=CONTOURING_FINISH($,$,'FINISH GENERAL REVOLUTION 1',30.000,$,#280,#61,#60,#130,#130,#132,0.0);

(* ***** Project ***** *)
#34=PROJECT('TURNING EXAMPLE 1',#35,(#1),$,$,$);
#35=WORKPLAN('MAIN WORKPLAN',(#36,#37),$,#52,$);
#38=MACHINING_WORKINGSTEP('WS ROUGH CIRCULAR_FACE 2',#56,#11,#22);

(* ***** Functions / Technology ***** *)
#60=TURNING_MACHINE_FUNCTIONS(.T.,$,$,(),.F.,$,$,(),$,$,$);
#61=TURNING_TECHNOLOGY($,.TCP.,#62,0.300,.F.,.F.,.F.,$);
#62=CONST_SPINDLE_SPEED(500);

(* ***** Strategies ***** *)
#131=UNIDIRECTIONAL_TURNING($,$,(3.000),$,$,$,$,$,2.000,$,$);
#139=CONTOUR_TURNING($,.F.,(0.500),$,$,$);

(* ***** Placements / Lengths ***** *)
#103=AXIS2_PLACEMENT_3D('SETUP 1',#104,#105,#106);
#104=CARTESIAN_POINT('SETUP1: LOCATION ',(0.000,0.000,0.000));
#105=DIRECTION(' AXIS ',(1.000,0.000,0.000));
#106=DIRECTION(' REF_DIRECTION',(0.000,0.000,1.000));

(* ***** Tools ***** *)
#280=TURNING_MACHINE_TOOL('',#281,(#283),120,40,$)
#281=GENERAL_TURNING_TOOL(#282,.LEFT.,40,60,.CW.);
#282=TOOL_DIMENSION($,$,$,$,25,5,7,3,5,0.5,$);
#283=CUTTING_COMPONENT(0.000000,$,$,$,$);

ENDSEC;
END-ISO-10303-21;
```

Gambar 3.1 Pembagian grup format ISO 14649

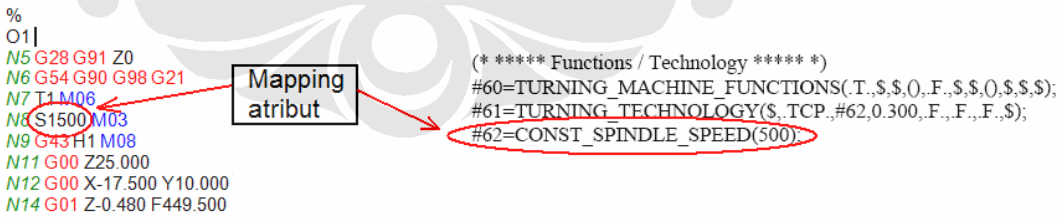
Dalam proses ini hal yang pertama kali dilakukan adalah menyusun aliran data struktur *file* STEP-NC sesuai dengan format standarnya dari ISO 10303, dan mencari informasi yang dibutuhkan untuk kasus *general revolution*. Metode *mapping* yang dilakukan adalah mengurutkan secara sekuensial standar ISO 14649 ini dari kelas yang terbesar menjadi kelas terkecil, beserta atributnya.

Kelas tertinggi yakni *Project* memuat data-data *workplan*, *machining workstep*. Kelas ini menjadi acuan untuk proses *breakdown* atau proses pemecahan entitas-entitas yang terkait dalam kasus *general revolution* dan yang diperlukan untuk konversi ini. Proses breakdown untuk kasus *general revolution* dapat dilihat pada gambar 3.3.

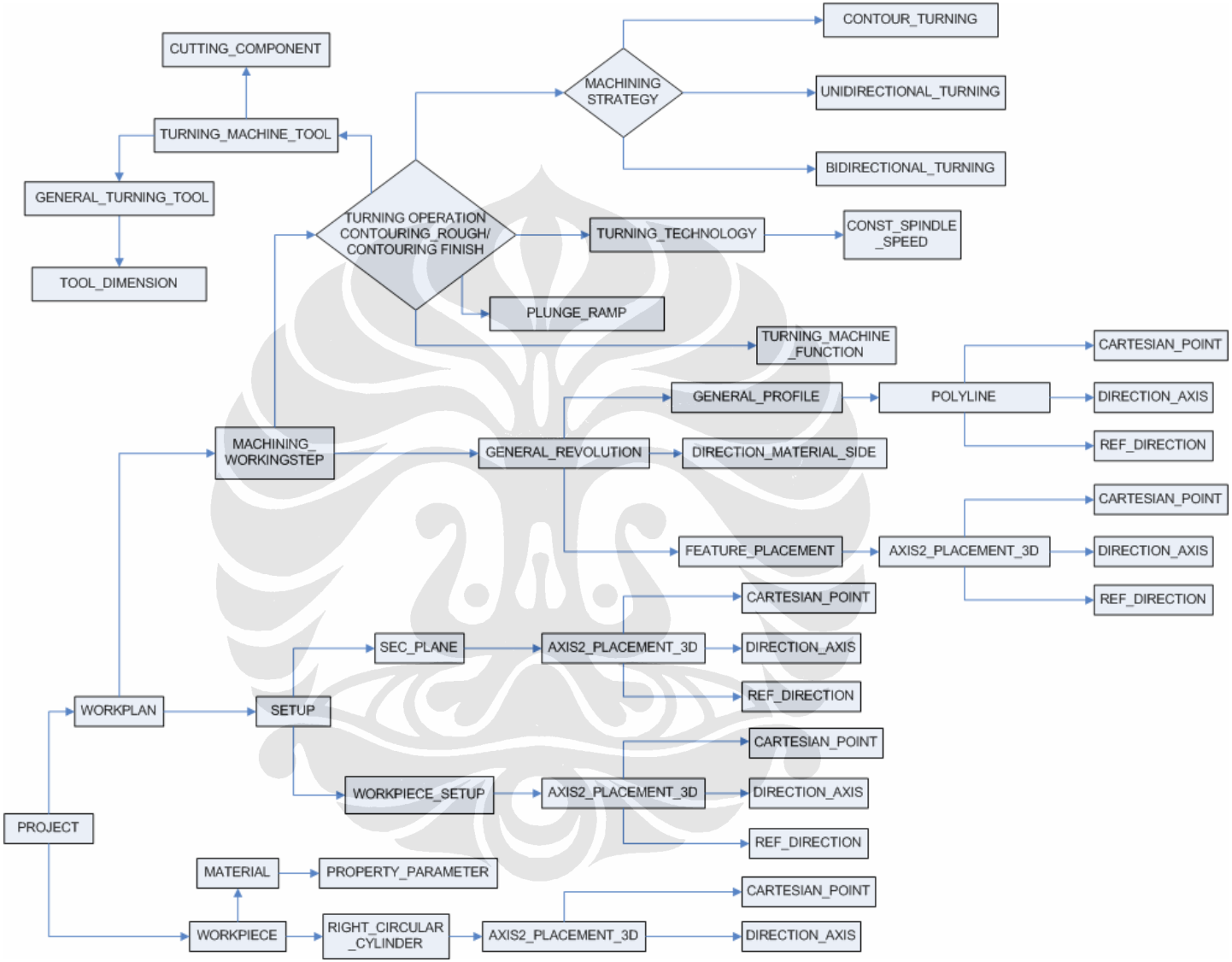
Setelah proses *breakdown* ini kemudian di dicari arti dari atribut-atribut yang melekat pada entitas, contohnya entitas *Const_Spindle_Speed* mengandung atribut yaitu :

CONST_SPINDLE_SPEED	
Rot_speed	Kecepatan konstan putaran spindle dalam unit revolusi/detik konstan

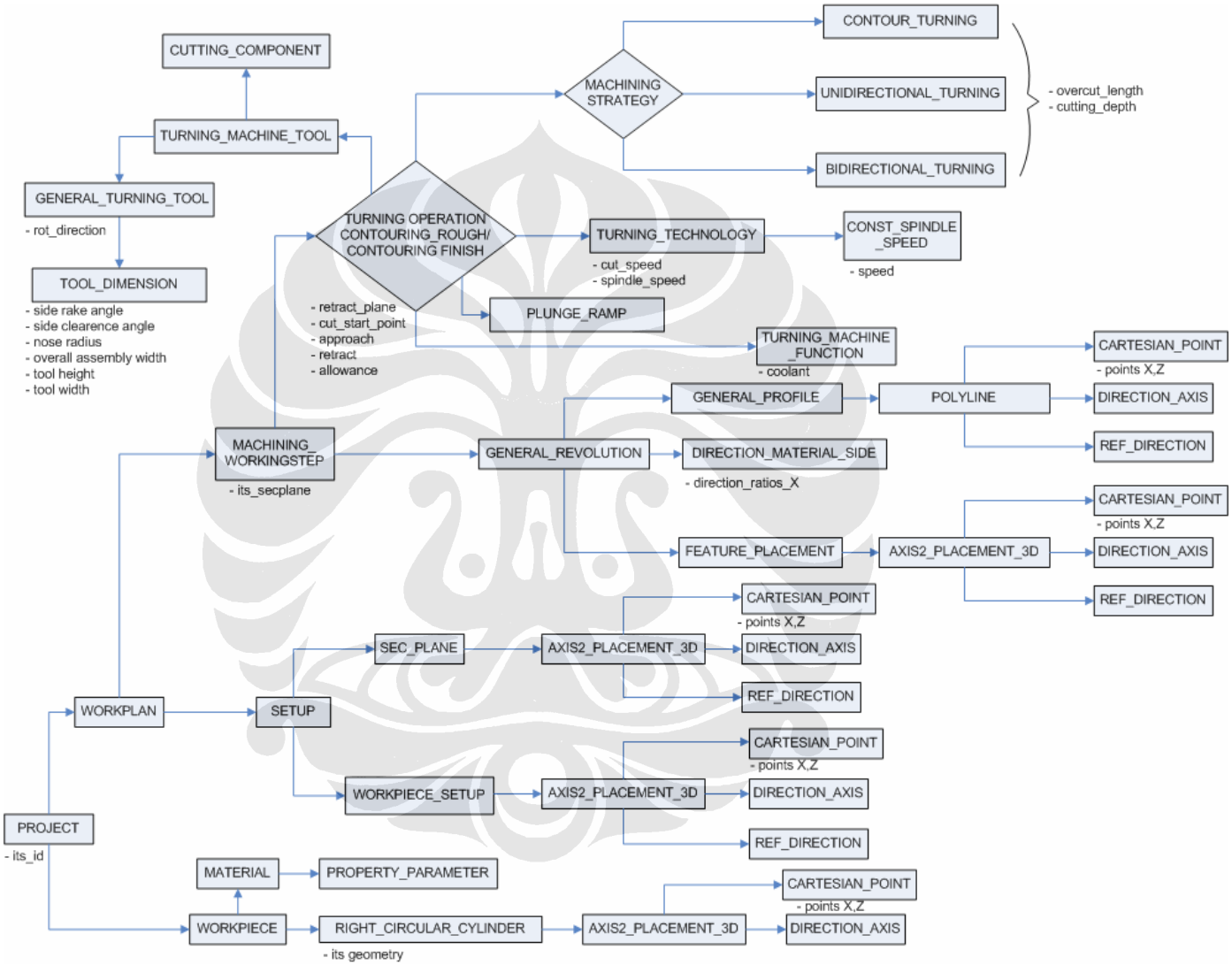
Dari pengertian atribut tersebut dapat diketahui bahwa atribut ini dapat dipakai untuk fungsi S pada format G-Code yaitu *spindle speed function word*. Perhatikan gambar 3.2 berikut.



Gambar 3.2 Proses *mapping* atribut



Gambar 3.3 Proses *breakdown* untuk kasus *general revolution*



Gambar 3.4 *Mapping* atribut

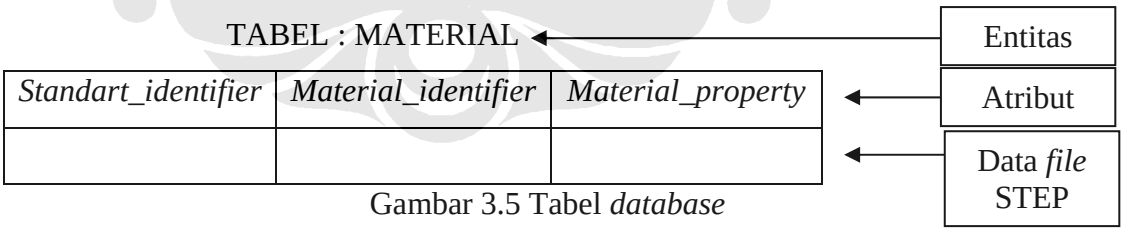
3.2 *DATABASE*

Proses dalam pembuatan suatu software konversi memerlukan suatu aplikasi yang dapat memuat data-data dan menyimpannya, serta data-data tersebut dapat diambil sewaktu-waktu atau sesuai yang diperlukan, maka daripada itu diperlukan suatu aplikasi *database*.

Database merupakan kumpulan data yang distrukturkan untuk memudahkan pemrosesan untuk menghasilkan suatu informasi. *Database* tersebut dikelola oleh suatu sistem manajemen *database* dan memakai bahasa universal yakni *Stuctured Query Language (SQL)*[6] dalam menjalankan perintah-perintahnya.

MySQL adalah program yang akan digunakan dalam penulisan ini. MySQL ini adalah peranti lunak yang menggunakan *database* relasi (*Relational Management Database System* atau RDBMS), seperti halnya ORACLE, PostgreSQL, Microsoft SQL dan lain sebagainya.

Penggunaan *database* dalam penulisan ini adalah dengan membuat tabel-tabel untuk setiap entitas-entitas yang sudah melalui proses *mapping* tersebut diatas. Sedangkan kolom-kolom dalam tabel tersebut diisi oleh atributnya masing-masing, sehingga nama tabel adalah nama entitas dan nama atributnya adalah nama kolom-kolomnya. Jumlah kolom sesuai dengan jumlah atribut masing-masing entitas. Berikut adalah contoh dari tabel dari *database* yang dibuat.



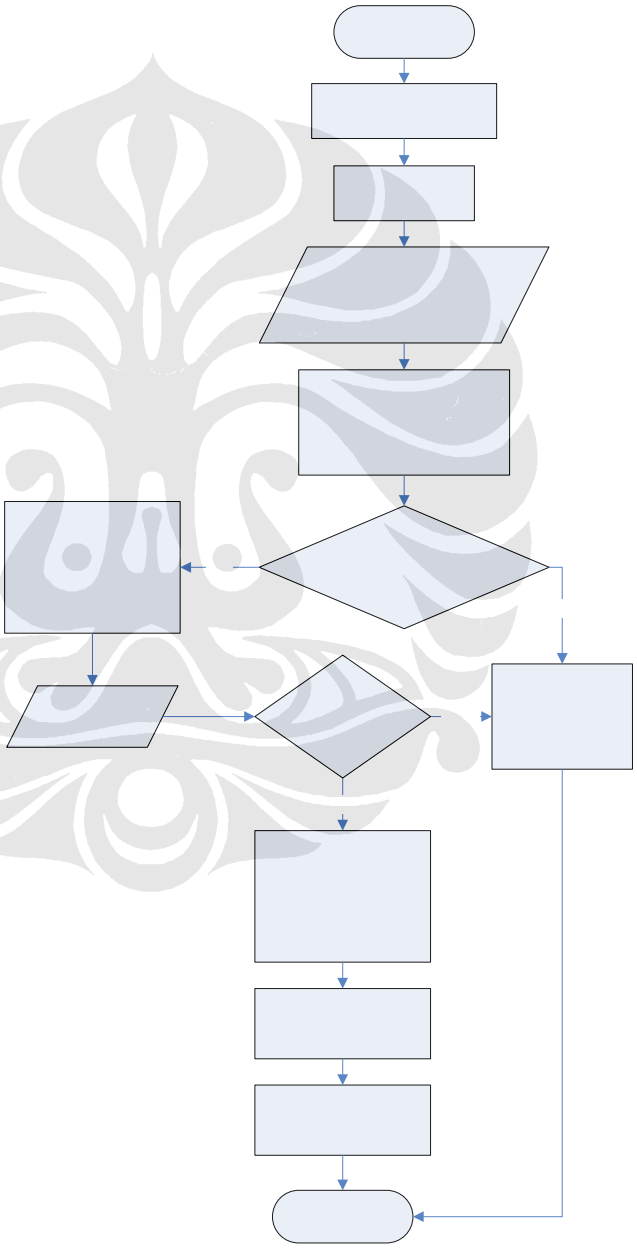
Gambar 3.5 Tabel *database*

Hubungan antar tabel dapat dilakukan dengan membuat relasi-relasi antar tabel dengan penggunaan satu buah *primary key* pada setiap tabelnya. Fungsinya relasi ini agar data dapat diakses oleh tabel lain dan informasi data membentuk satu kesatuan dimana hal ini sesuai dengan format *file* STEP-NC dimana satu

entitas akan selalu berhubungan dengan entitas lain. Relasi tabel dapat dilihat sesuai dengan hasil *mapping* untuk kasus *general revolution* ini.

3.3 MERANCANG *SOFTWARE* KONVERSI

Tahapan-tahapan untuk *software* untuk mengkonversi *file* setelah membangun *database* seperti terlihat pada gambar 3.6 berikut :



Gambar 3.6 Tahapan *software* konversi

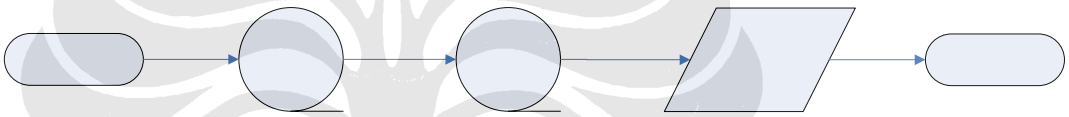
Dua tahapan utama untuk proses konversi adalah[7] :

1. Tahapan *Search and Send* Data
2. Tahapan *Take and Processing* Data

3.3.1 *Search and Send Data*

Proses penelusuran entitas dan atributnya yang akan dikirim ke *database* ini mengacu pada proses *mapping* yang telah dilalui. Proses dimana mencari entitas yang utama beserta atributnya sampai kepada cabang entitas yang paling terakhir.

Berikut adalah urutan proses *search and send* data :



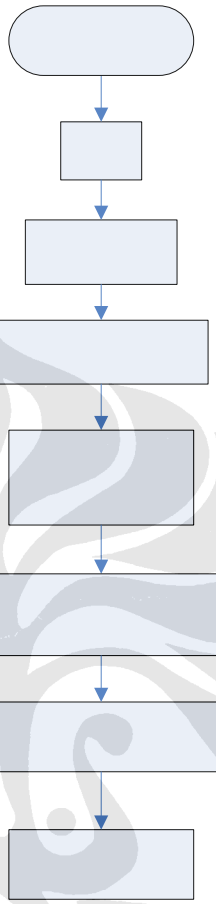
Gambar 3.7 Urutan proses *search and send* data

3.3.2 *Take and Processing Data*

Pengambilan data atribut yang diperlukan dari *database* sesuai dengan *rule* program dan mencetaknya dalam format G-code adalah fungsi dari tahapan ini. Tahapan ini mengacu pada *rule* take and processing data, *rule* utama *take and processing data* dibagi dalam 3 block sesuai dengan badan format G-Code. Block tersebut adalah :

1. *Rule Block Start*
2. *Rule Command*
3. *Rule Block End*

3.3.2.1 *Rule Block Start*



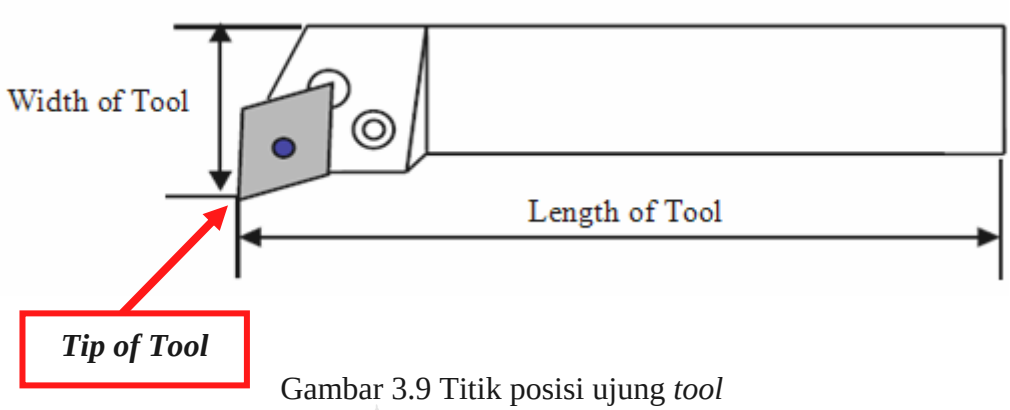
Gambar 3.8 Struktur *Block Start*

Aturan pertama yang diberlakukan dalam format G-Code adalah aturan untuk menyalakan fungsi-fungsi pemesinan, seperti *tool* yang akan dipakai, kecepatan dan arah putaran spindel, serta menyalakan *coolant* apabila diperlukan.

3.3.2.2 *Rule Command*

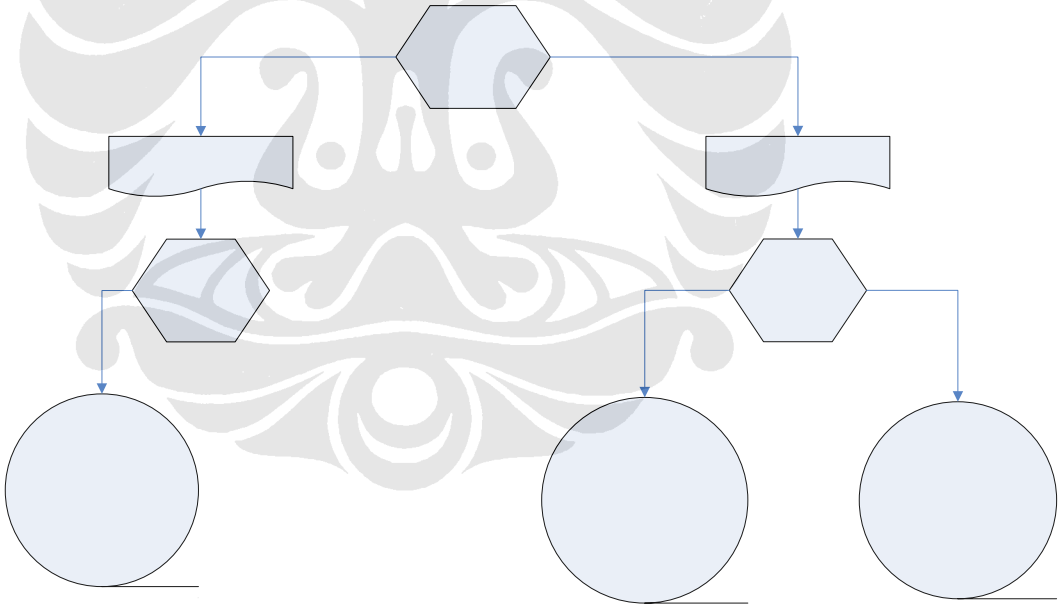
Rule Command adalah blok yang memuat *rule* pergerakan *tool* proses pemakanan. Pada *rule* ini untuk menentukan startegi yang akan digunakan dengan proses operasi yang akan dilakukan, apakah itu proses *roughing* saja atau proses *roughing* dan *finishing*.

Hal yang perlu diingat posisi *tool* dalam operasi untuk *turning* adalah bahwa titik/posisi *tool* (x dan z) berada pada titik ujung *tool*.



Gambar 3.9 Titik posisi ujung *tool*

Pada penulisan ini untuk kasus *General_Revolution* hanya terdapat satu strategi untuk *roughing* yakni strategi *Unidirectional_Turning* sedangkan untuk *finishing* terdapat dua strategi yaitu *Unidirectional_Turning* dan *Contour_Turning*.



Gambar 3.10 *Operation*

Sesuai dengan tipe operasi yang menjadi pilihan untuk *general_revolution* ini maka akan terdapatnya 3 buah *rule* yang menjadi basis dalam algoritma

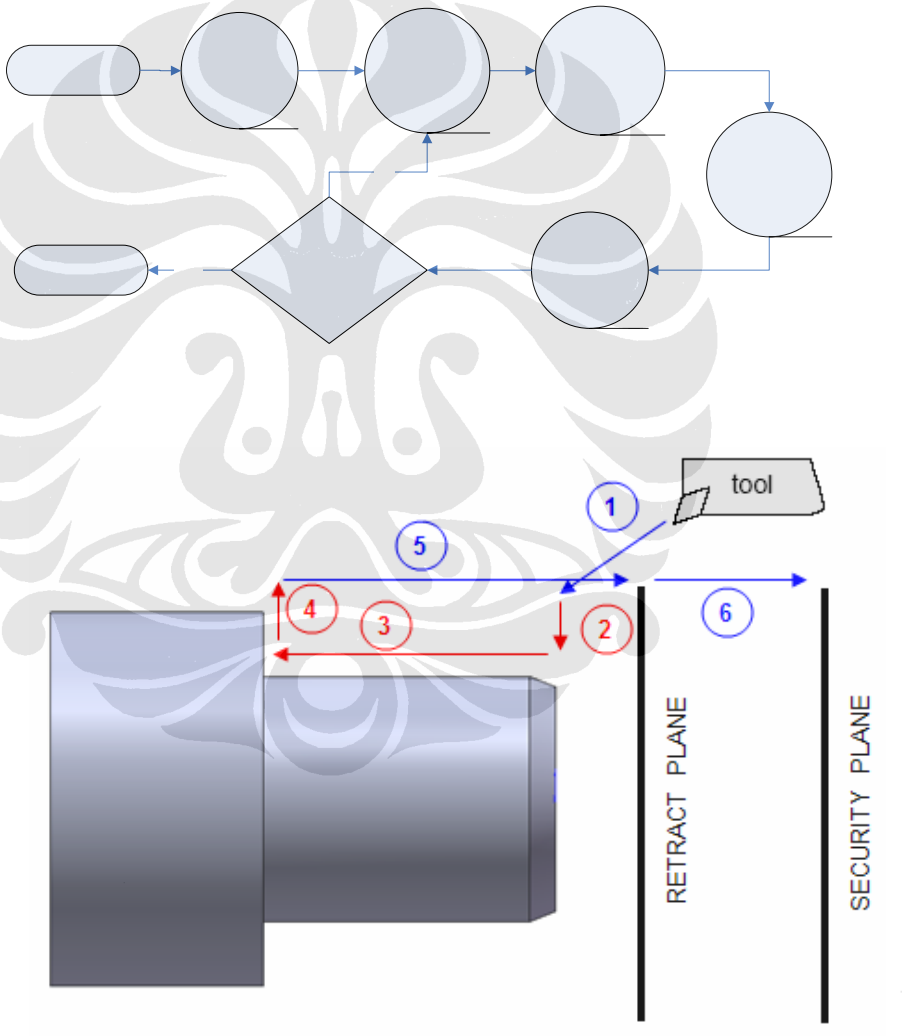
pemogramannya. Apabila ada proses lain seperti *bidirectional_turning* maka dalam program ini tidak dimasukkan jenis operasi tersebut.

Pada tahapan-tahapan proses dari ketiga *rule* operasi diatas terdapat perbedaan dalam cara pendekatan *tool*, dan karakteristik pergerakannya.

1. *Unidirectional_Turning*
2. *Contour_Turning*

3.3.2.2.1 *Roughing* dan *Finishing Unidirectional_Turning*

Secara umum tahapan strategi ada 6 tahap yakni :

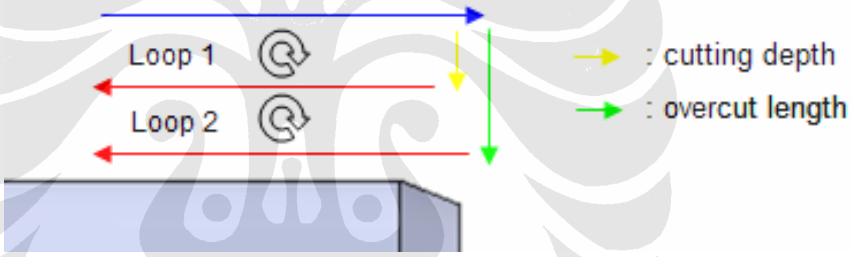


Gambar 3.11 Strategi *Unidirectional_Turning*

1. *Approach* adalah tahapan dimana *tool* mendekati benda kerja. Ada dua kemungkinan dalam pendekatan *tool* menuju benda kerja yakni pertama

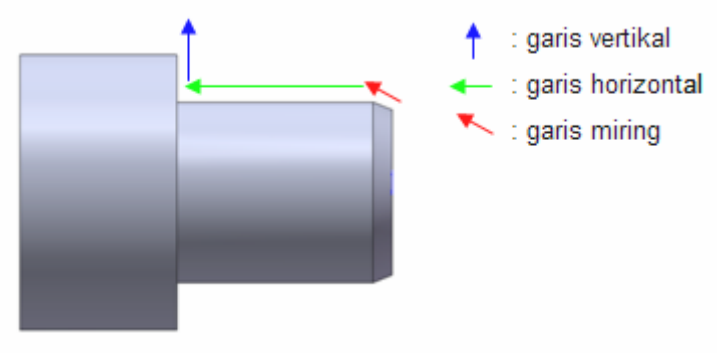
tool mendekati dari titik nol mesin sampai dimensi material awal. Apabila data material geometri *file* STEP berisi nilai optional ($) maka data untuk pendekatan diambil dari dimensi terbesar dari arah X dan Z-nya fitur/profil yang akan dibentuknya dengan syarat adanya penambahan *allowance* agar *tool* dengan parameter G00 berhenti sebelum menabrak material.

2. Tahapan kedua adalah *radial cutting* yaitu pemakanan arah tegak lurus dari sumbu benda kerja, besarnya kedalaman ditentukan oleh besarnya nilai *cutting_depth* pada proses. *Radial cutting* untuk strategi *unidirectional_turning* ada dua macam adalah yaitu *radial cutting* awal yang hanya memperhitungkan *cutting depth* saja, sedangkan *radial cutting* kedua setelah satu *loop* pemakanan selesai akan memperhitungkan *cutting overcut length* untuk mencapai posisi yang dimaksudkan.



Gambar 3.12 Tahapan *Radial cutting* awal dan proses *radial cutting* berikutnya

3. *Axial Cutting* merupakan tahap berikutnya. Proses pemakanan axial ini adalah pemakanan searah dengan sumbu benda kerja. Dalam proses pemakanan ini akan mengikuti bentuk dari benda yang akan dikerjakan. Dalam penulisan ini terdapat tiga buah profil yaitu garis vertikal (garis tegak lurus sumbu), garis horizontal (garis searah sumbu), dan garis miring.

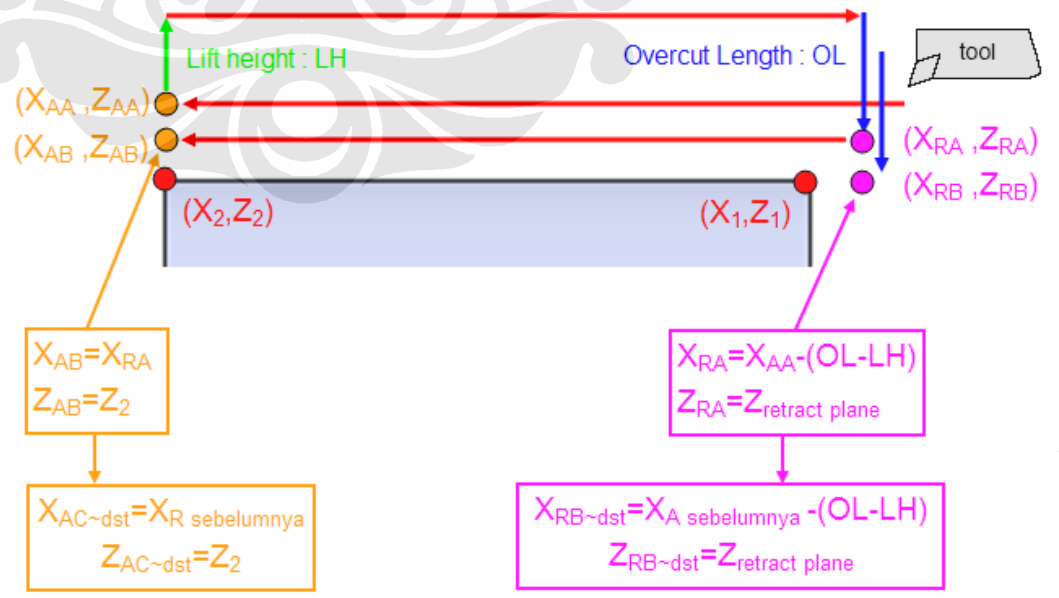


Gambar 3.13 Profil *General revolution*

Penentuan sebuah garis pada format *file* STEP dapat dilihat dengan titik-titik yang sudah dimasukkan pada *database* dengan identifikasinya *POLYLINE*. Apabila start point sebuah *polyline* memiliki nilai X yang sama dengan nilai X pada end pointnya maka program akan mengidentifikasikan sebagai sebuah garis vertikal, apabila Z-nya yang sama maka diidentifikasi sebagai garis horizontal, sedangkan bila kedua nilai X dan Z-nya berbeda antar start point dan end point maka dinyatakan sebagai garis miring.

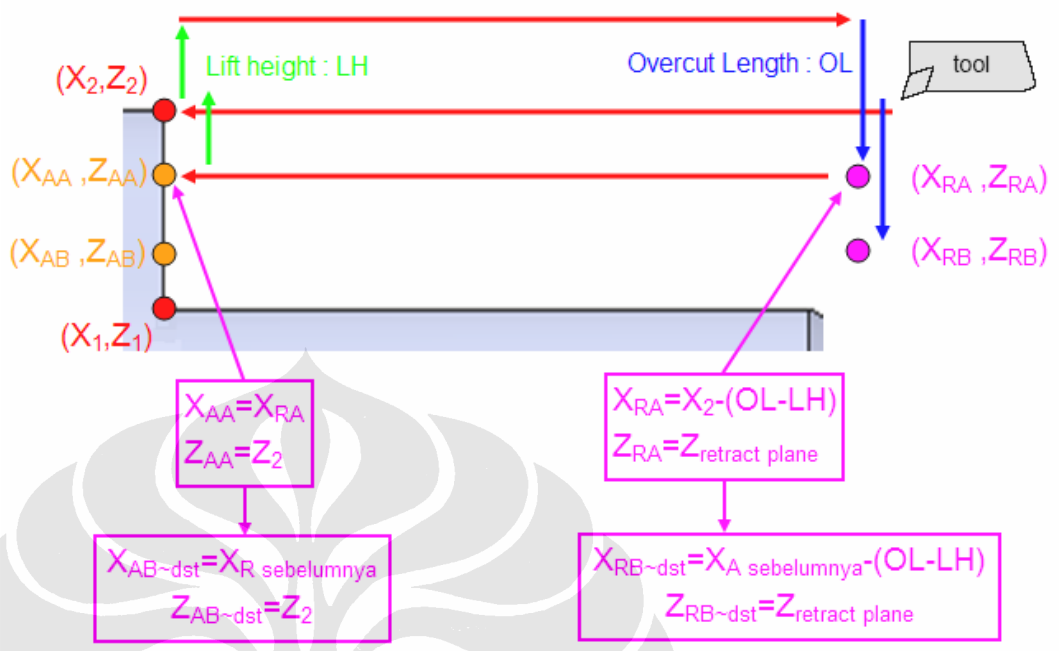
Adapun aturan untuk setiap *polyline* untuk dijalankan oleh program akan berbeda. Aturan untuk setiap *polyline* dapat dilihat dibawah ini :

Polyline Horizontal :



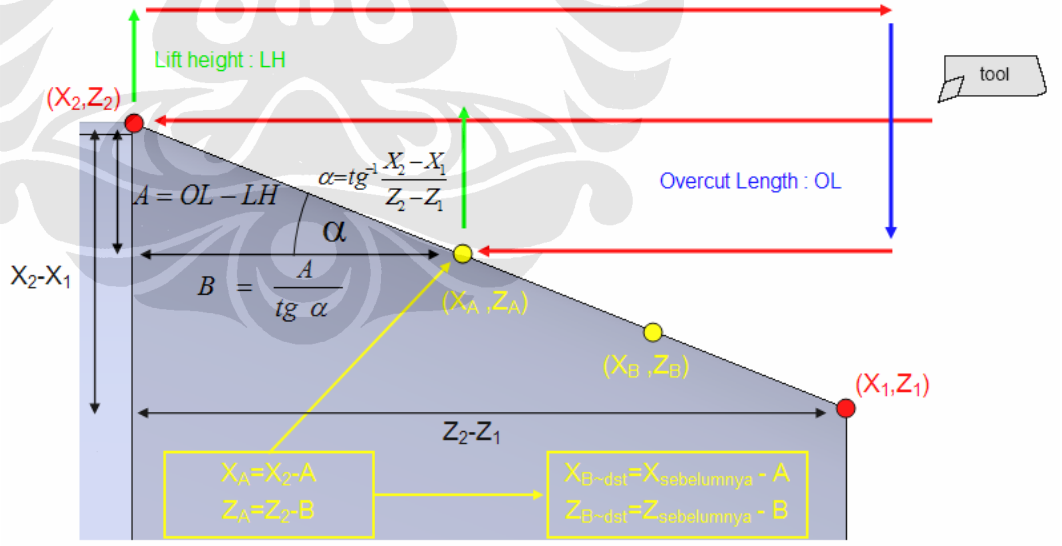
Gambar 3.14 Aturan *Polyline* Horizontal

Polyline Vertikal :



Gambar 3.15 Aturan *Polyline* Vertikal

Polyline Miring :



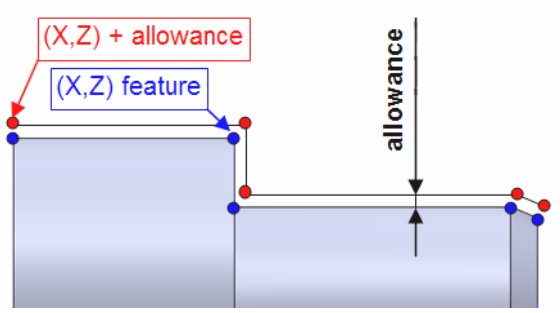
Gambar 3.16 Aturan *Polyline* Miring

4. *Lift Height* yang berupa gerakan menarik kembali ke daerah "bebas" setelah melakukan proses pemotongan, perlu mempertimbangkan beberapa hal, yaitu apakah pada saat tarik kembali tidak menabrak benda

kerja, melihat jenis pergerakannya yakni tegak lurus sumbu atau mempunyai suatu sudut saat *tool* mencapai titik terakhir pemakanan axial. Pergerakan *lift height* ini dapat dilihat pada entitas strategi-nya berupa atribut *lift_direction*. Walaupun demikian parameter yang digunakan adalah G01 untuk *lift_height* ini.

5. Dalam sebuah *loop* pemakanan akan diakhiri oleh proses *retract*, yakni proses penarikan kembali *tool* pada posisi semula untuk memulai kembali sebuah *loop* pemakanan atau mengakhiri seluruh proses. Nilai dari *retract* diambil dari entitas *contour*ing dengan mengambil nilai dari atribut *retract_plane* ataupun sesuai dengan ketentuan pada ISO 14649-10 mengenai *machining*_operation bahwa apabila atribut *retract_plane* tidak ada nilainya maka NC controller menentukan sendiri *retract_plane* tersebut atau dapat mengambil nilai *retract* dari strategi yang akan digunakan. Untuk strategi ini maka referensi pengambilan nilai *retract_plane* dapat diambil dari entitas cartesian_point dengan *identifier feature placement* dengan mengambil nilai Z-nya.
6. Tahap ini memperhitungan apakah dimensi *general_revolution* telah tercapai atau tidak, apabila dimensi belum tercapai maka *tool* akan kembali ke tahapan 2 dan terus-menerus melakukan *looping* sampai seluruh dimensi tercapai. Apabila seluruh dimensi tercapai maka pergerakan *tool* selanjutnya ke posisi bidang aman (*security_plane*) atau dengan posisi x dan y origin mesin.

Perbedaan antara *roughing* dan *finishing* pada strategi ini adalah *allowance* side. Besarnya nilai *allowance* side dapat dilihat pada entitas *contour*ing masing-masing yaitu entitas *contouring_rough* atau *contouring_finish*. Nilai *allowance* ini diperhitungkan sebelum melakukan proses pemakanan, nilai ditambahkan pada dimensi X dan Z tiap *polyline* yang terdapat pada fitur *general_revolution*.

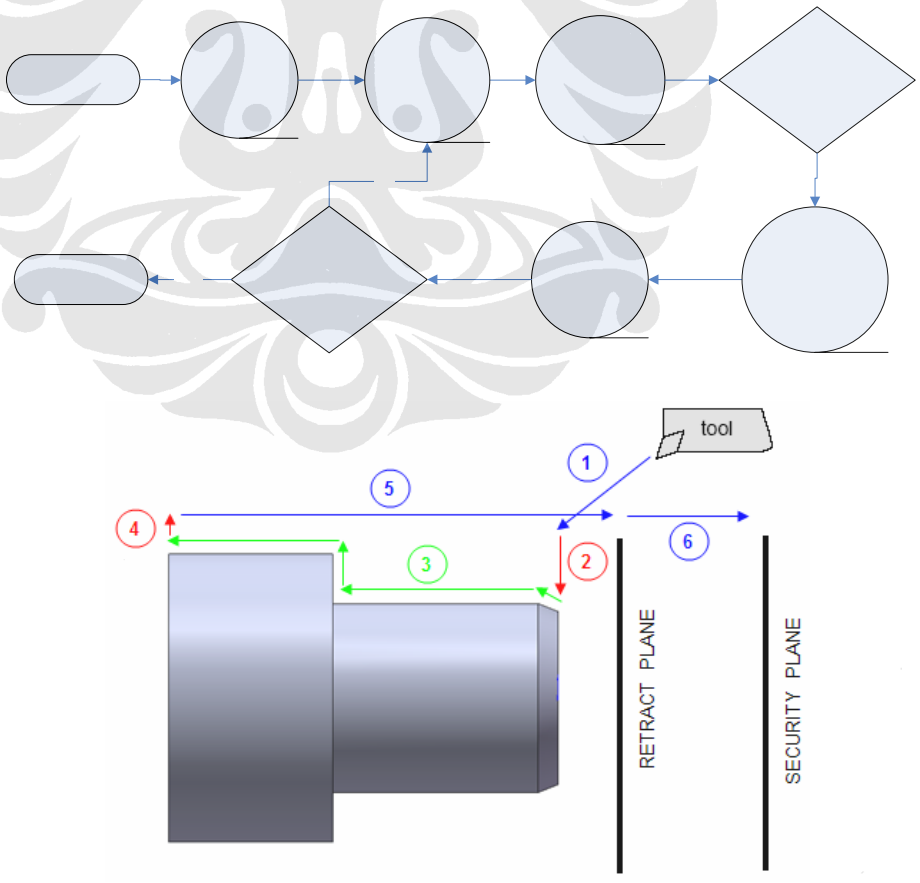


Gambar 3.17 Profil *allowance*

3.3.2.2.2 *Finishing Contour_Turning*

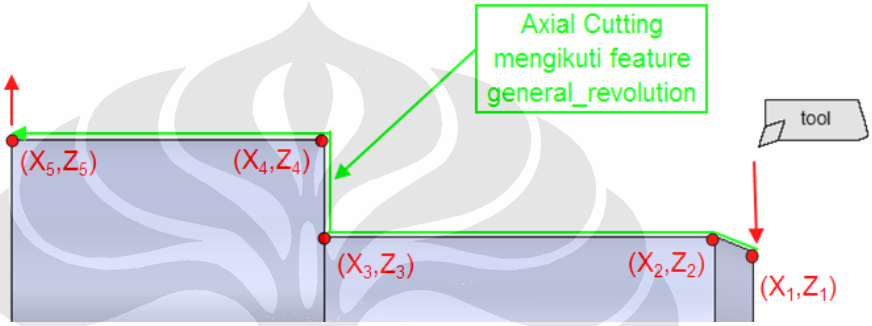
Sesuai dengan ISO 14649, strategi *contour_turning* dikhususkan untuk proses *finishing* saja. Secara umum strategi ini mempunyai banyak persamaan dengan strategi *unidirectional_turning*. Seperti halnya untuk tahapan *approach*, *radial cutting*, *lift height*, serta *retract* mempunyai persamaan proses.

Adapun perbedaan antara strategi *unidirectional_turning* dan *contour_turning* dapat dilihat pada gambar dibawah ini :



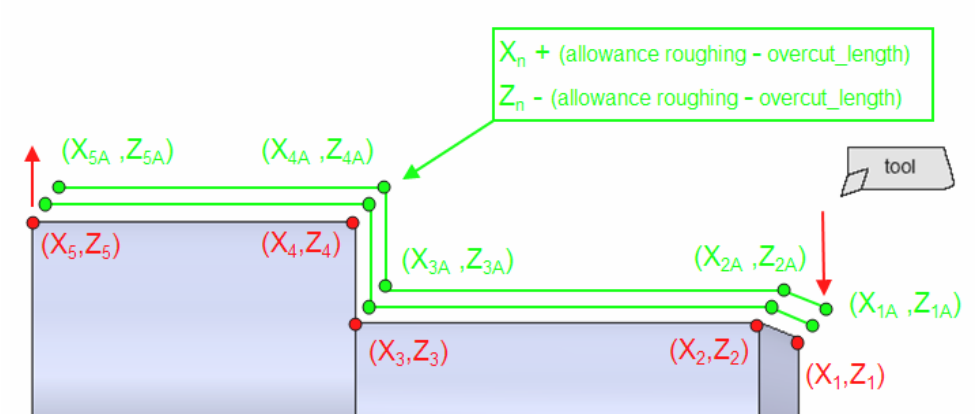
Gambar 3.18 Strategi *Contour_Turning*

Perbedaannya terdapat pada proses *axial cutting* dimana strategi *unidirectional_turning* akan mengindentifikasi jenis *polyline* tersebut dan akan mengeksekusi aturan yang diberlakukan pada *polyline* tersebut, sedangkan untuk strategi *contour_turning* pada proses axial cutting tersebut akan mengindentifikasikan seluruh *polyline* menjadi suatu kesatuan yang tak terpisahkan. Sehingga menjadi satu kali pemakanan untuk keseluruhan fitur pada benda kerja tersebut.



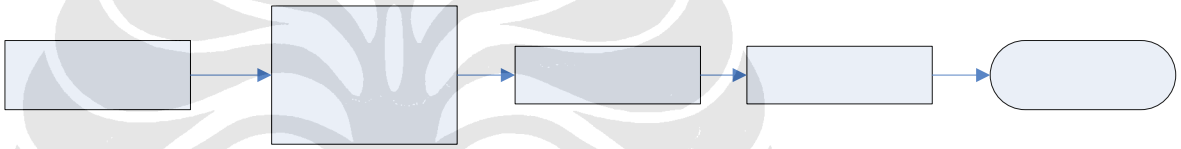
Gambar 3.19 *Axial cutting* strategi *contour_turning*

Walaupun pada strategi ini umumnya hanya satu kali pemakanan saja, tetapi tidak menutup kemungkinan terjadi lebih dari satu kali pemakanan. Apabila hal tersebut terjadi maka pada tahapan terakhir apabila dimensi fitur belum tercapai maka akan terjadi proses *looping* pemakanan kembali. Untuk mengetahui berapa *looping* yang terjadi dapat diketahui dari atribut *overcut_length* pada entitas *contour turning*, bilamana sisa *allowance* pada proses *roughing* lebih besar daripada overcut_length dari entitas *contour_turning* maka akan terjadi lebih dari satu kali *axial_cutting* dan dimensi tiap *polyline* akan disesuaikan dengan ditambahkannya sisa *allowance* yang belum terkena proses pemotongan.



Gambar 3.20 Nilai X,Y pada *looping axial cutting contour_turning*

3.3.2.3 *Rule Block End*



Gambar 3.21 Struktur *Block End*

Untuk *Block End* beberapa hal yang harus diperhatikan untuk semua jenis proses *turning* adalah me-nonaktifkan beberapa fungsi yang terdapat pada *block start* seperti halnya mematikan status *coolant* ataupun memberhentikan putaran spindel.