Chapter 6

A general theory of singularities

The singularity theory is aimed at denotation, extension, and truth – not just one of these notions. So the theory is pitched at a sufficiently general level to deal with each of them in exactly the same way. Let *L* be a fragment of English free of the semantic terms 'denotes', 'extension', 'true', ‘false’, 'true of', ‘false of’. This is the non-semantic part of English. We obtain the language ₤ by adding to *L* one of these semantic terms: ₤ is *L*+‘extension’, or *L*+‘true’/‘false’, or *L*+ ‘true of’/‘false of’. The singularity theory goes on to address the associated paradoxes by the identification of singularities. It doesn’t matter which term is added – the account is just the same. In fact, the theory goes forward with a schematic term *t* standing in for any of these semantic predicates – any one of them can be plugged in for *t*. The theory aims at a unified account of the semantic paradoxes.

In the course of this chapter, I’ll be returning to the pathological expressions C (‘the sum of the numbers denoted by expressions on the board in room 213’), P (‘non-self-membered extension of a predicate on the board in room 213’), L (‘The sentence written on the board in room 213 is not true’) and H (‘predicate on the board in room 213 not true of itself’).

6.1 Preliminaries

I turn first to two preliminary matters. The first concerns the notion of a reflective context. As we’ve seen in previous chapters, a context can be *explicitly* reflective: for example, the stock-taking and re-evaluation stages of the *repetition* discourse about L are reflective with respect to L. But, as we've already observed, a context can be reflective without being explicitly so. For example, if you innocently say "'Snow is white' is true", your context of use is taken to be reflective with respect to L – L counts as a truth in your context of use, as it does when assessed in an explicitly reflective context. In this section, I'll say more about reflective contexts of both kinds. The second preliminary matter concerns determination sets and the semantic value of expressions.

6.1.1 Explicitly reflective contexts

In developing the formal theory, we are interested in semantic and contextual aspects. On the semantic side, given an expression σ, we are interested in the expressions to which σ makes reference, and the expressions to which those expressions make reference, and so on – we're interested in the whole network of expressions generated from σ. On the contextual side, we are concerned with the context in which the expression occurs, and it is a crucial matter as to whether or not the context is explicitly reflective.

We've already seen that a context can be non-explicitly reflective without being explicitly reflective. In the other direction, a context can be explicitly reflective without being non-explicitly reflective. For example, this is so in any case of a repetition. For example, C and C* determine exactly similar semantic networks, but while C*'s context is explicitly reflective with respect to C, C's context is not (similarly with P and P*, and L and L*). A context can explicitly fail to be reflective – for example, when we intentionally produce a pathological phrase. And it's evident that a context can non-explicitly fail to be reflective – for example, when I unwittingly write C on the board.

Repetitions are not the only examples where a context is explicitly reflective without being non-explicitly reflective. For another kind of example, consider the following infinite truth chain:

(1) 2 is not true.

(2) 3 is true

.
.
.

(2n-1) 2n is not true

(2n) 2n+1 is true.

.
.
.

And consider a further sentence:

(0) 1 is true.

By Symmetry, we treat 0 as a pathological member of an enlarged chain. Its pathology is indicated by the single infinite branch that composes its primary tree.

Now contrast 0 with the sentence 0^+, produced through strengthened reasoning as follows:

> 2 heads a chain, and so 2 is pathological. Since 2 is pathological, 2 is not true. That's what 1 says; so
> (0^+) 1 is true.

0^+ is not pathological – it's true. How can we capture the difference between 0 and 0^+? The difference between them is a matter of the different contexts in which they are produced. The context of (0^+) is *explicitly reflective* with respect to 1, 2, and all the members of the chain. 0^+ occurs in a context in which it is part of the common ground that 1, 2, … are pathological. We will be unable to capture the difference between 0 and

0^+ if we ignore this contextual difference. As we've seen in previous chapters, a parallel point can be made about repetitions: if we ignore the contextual differences between C and C* (and P and P*, and L and L*), we will be unable to capture the difference between a repetition and the original pathological expression.

Suppose we did ignore the contextual difference between 0 and 0^+. The primary representation of 0 is <type(0),c_0,c_0>, and the primary representation of 0^+ is (type(0^+),c_{0+},c_{0+}>, where 'c_0' stands for the context of 0, 'c_{0+}' for the context of 0^+, and 'c_n' stands for the context of (n), for n>0, Then it is easy to see that the (unfounded) primary trees for 0 and 0^+ will be exactly similar. The primary tree for 0 is:

$$\begin{array}{c} \langle \mathrm{type}(0), c_0, c_0 \rangle \\ | \\ \langle \mathrm{type}(1), c_1, c_0 \rangle \\ | \\ \langle \mathrm{type}(2), c_2, c_1 \rangle \\ . \\ . \\ . \end{array}$$

The primary tree for 0^+ is similar:

$$\begin{array}{c} \langle \mathrm{type}(0^+), c_{0+}, c_{0+} \rangle \\ | \\ \langle \mathrm{type}(1), c_1, c_{0+} \rangle \\ | \\ \langle \mathrm{type}(2), c_2, c_1 \rangle \\ . \\ . \\ . \end{array}$$

Without incorporating further information about the explicitly reflective nature of the context c_{0+}, we cannot adequately distinguish 0 and 0^+.[1]

So we need to flag explicitly reflective contexts in some way. Here is one way to proceed. To fix ideas, consider the case of the rehabilitation of C. As we saw in 2.5, we can *rehabilitate* C – in a suitably reflective context, we can say:

(†) The number denoted by C is π+6.

Let D be the definite description token composed of the first five words of (†). The primary representation of D is <type(D),c_D,c_D>. The second element indicates that the occurrence of 'denotes' in D is represented by 'denotes$_{cD}$', where this representation applies to D and any coextensive token of the same type. We call the associated schema the c_D-schema – the occurrence of 'denotes' on the left hand side of the c_D-schema is coextensive with the occurrence of 'denotes' in D. So the second element indicates that C (the only member of D's determination set) is to be assessed by the c_D-schema. The third element indicates the schema by which D itself is to be assessed, as fixed by D's context. All it takes to assess D in its context is the application of the c_D-schema to C, so the evaluating schema for D is the c_D-schema. In the case of D, then, the schemas indicated by the second and third elements are the same. Now both these schemas are explicitly reflective with respect to C – in D's context, it is part of the common ground that C is pathological and does not denote (that is, does not denote$_{cC}$). We could flag this by adding subscripts to the second and third entries of D's primary representation: <type(D),$[c_D]_C$,$[c_D]_C$>. By this notation we understand that the schemas indicated by the second and third entries are reflective with respect to C.

This adjustment to D's primary representation generates corresponding adjustments to D's primary tree. The resulting tree looks like this:

$$\begin{tikzcd}[column sep=small]
 & \langle \mathrm{type}(D),[c_D]_C,[c_D]_C\rangle \arrow[d, dash] & & \\
 & \langle \mathrm{type}(C),c_C,[c_D]_C\rangle \arrow[d, dash] & & \\
 & \langle \mathrm{type}(C),c_C,c_C\rangle \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & & \\
\mathrm{type}(A) & \mathrm{type}(B) & \langle \mathrm{type}(C),c_C,c_C\rangle \arrow[dl, dash] \arrow[d, dash] & \\
 & \mathrm{type}(A) & \mathrm{type}(B) & \ddots
\end{tikzcd}$$

So we have imported into D’s primary tree information about those schemas that are explicitly reflective with respect to C.

Notice that the second node is a secondary representation of C, according to which C is evaluated by a schema that is reflective with respect to C. This is really what we want to capture – that C is reflectively evaluated. So we will rewrite this node as $\langle \mathrm{type}(C),c_C,r_C\rangle$. For the purposes of the formal theory, what matters is that the evaluating schema is reflective with respect to C, and we can abstract away from the particular context c_D. We can further simplify the tree by erasing the additional subscripts at the previous node – we have no need of them now. So the primary tree for D now looks like this:

$$\begin{tikzcd}[column sep=small]
 & \langle \mathrm{type}(D),c_D,c_D\rangle \arrow[d, dash] & & \\
 & \langle \mathrm{type}(C),c_C,r_C\rangle \arrow[d, dash] & & \\
 & \langle \mathrm{type}(C),c_C,c_C\rangle \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & & \\
\mathrm{type}(A) & \mathrm{type}(B) & \langle \mathrm{type}(C),c_C,c_C\rangle \arrow[dl, dash] \arrow[d, dash] & \\
 & \mathrm{type}(A) & \mathrm{type}(B) & \ddots
\end{tikzcd}$$

We can follow a similar procedure in the case of 0^+. Here, the context of 0^+ is explicitly reflective to all the sentences in the chain: 1, 2, … . So the additional subscript will indicate the set {1,2,…} – the set of all sentences with respect to which c_{0*} is explicitly reflective. (In the case of D, as a matter of convenience, the subscript indicated C, rather than the unit set {C}.) We adjust the primary tree for 0* to obtain:

$$\begin{tikzcd}[row sep=small]
\langle \mathrm{type}(0^+),[c_{0+}]_{\{1,\,2,\,3,\ldots\}},[c_{0+}]_{\{1,\,2,\,3,\ldots\}}\rangle \arrow[d, dash] \\
\langle \mathrm{type}(1),c_1,[c_{0+}]_{\{1,\,2,\,3,\ldots\}}\rangle \arrow[d, dash] \\
\langle \mathrm{type}(2),c_2,c_1\rangle \\
\vdots
\end{tikzcd}$$

The second node is a secondary representation of (1), according to which (1) is evaluated by a schema reflective with respect to (1) (and to (2), (3), …). So we now replace $\langle \mathrm{type}(1),c_1,[c_{0+}]_{\{1,\,2,\,3,\ldots\}}\rangle$ by $\langle \mathrm{type}(1),c_1,r_1\rangle$, and erase all additional subscripts appearing in higher nodes. The primary tree now looks like this:

$$\begin{tikzcd}[row sep=small]
\langle \mathrm{type}(0^+),c_{0+},c_{0+}\rangle \arrow[d, dash] \\
\langle \mathrm{type}(1),c_1,r_1\rangle \arrow[d, dash] \\
\langle \mathrm{type}(2),c_2,c_1\rangle \\
\vdots
\end{tikzcd}$$

We now have three versions of the primary trees for D and O^+: the preliminary tree, the tree with additional subscripts, and the tree with a node of the form r_α. For the purposes of the formal theory, the last version is the one we want. From now on, the 'official' primary tree for an expression σ will contain nodes of the form r_α wherever appropriate. For the purposes of the formal theory, we assume that all the facts about explicitly reflective contexts are in. Primary trees are taken to incorporate all relevant information about explicitly reflective contexts and explicitly reflective schemas.

In general, given an expression σ, we obtain its official primary tree through the following procedure. Start with its initial primary tree. If the second or third element of any triple indicates a schema that is explicitly reflective, add an additional subscript to that element to indicate the set of expressions to which the schema is explicitly reflective. Identify any secondary representation, of an expression ρ, of the form $<\text{type}(\rho),c\rho,[c\tau]_U>$, where ρ is a member of the set U. Replace that representation by the triple $<\text{type}(\rho),c\rho,r\rho>$, and remove the additional subscript 'U' from all higher nodes.

6.1.2 Determination sets and values

As we said at the outset of this chapter, we start with a language *L* that is free of the terms 'denotes', 'extension', 'true' or 'true of', and then move to a language £, which is the result of adding one of these terms to *L*. The kind of pathology in which we are centrally interested will not arise in *L*. However, that is not to say that there won't be problematic expressions in *L*. For example, we find in *L* grammatically well-formed denoting phrases that fail to denote, such as 'the integer between 6 and 7'. This in turn leads to problematic expressions of £ - for example,

(N) the number denoted by 'the integer between 6 and 7'.

Even though the sole member of N's determination set is an expression of *L*, still no denotation can be determined for N. The kind of pathology here is different from that exhibited by, for example, C. There the problem is one of circularity or ungroundedness, where in trying to evaluate C, we are constantly led to expressions containing 'denotes'. The problem with N is not one of ungroundedness - we are led to an expression of *L* free of 'denotes'. The problem instead is that the sole member of N's determination set flatly fails to refer. We want the formal theory to elucidate the pathology exhibited by C, not that exhibited by N. Still, the formal theory must allow for phrases like N that fail to refer even though their determination sets are composed only of expressions of *L*.

Conversely, there are expressions of ₤ that are intuitively ungrounded, but which nevertheless have a value. For example, there are expressions of ₤ that are members of their own determination sets, and yet a value can be determined for them solely in terms of expressions of *L*. In more general terms, a value for an expression may be determined by a proper subset of its determination set. Suppose I write on the board the following predicates:

(O) moon of the Earth

(Q) extension of a predicate on the board with at least one member.

The predicate Q is a member of its own determination set, and is in this sense ungrounded. However, it is highly plausible that Q has a definite extension. Given the property that Q denotes, and given the empirical circumstances, the extension of O is a member of Q. And then the extension of Q has at least one member - and so the extension of Q has the property denoted by Q and so is a self-member. That is, Q has a definite extension: its two members are the extension of O and the extension of Q. The extension of Q is determined given the property that Q denotes, and the fact that the predicate O has a member. Though both O and Q are in the determination set for Q, we need only consider O's extension in order to fully determine a value for Q. We need determine only the extension of a predicate of *L* in order to determine an extension for Q.

Similar examples can be given for our other terms. If I write on the board the phrases "zero" and "the smallest natural number denoted by an expression on the board", then it is plausible that the latter denotes 0, even though it belongs to its own determination set. Or consider the liarlike sentence

(S) Snow is white or S is false.

One might suppose that S is true in virtue of the first disjunct - and if we do suppose this, then a truth value for the sentence as a whole can be determined just because a truth value for a sentence of *L* can be determined. We cannot reasonably expect the formal theory itself to elucidate such cases - in general, they are too dependent on the empirical (or mathematical) circumstances, and not dependent enough on intrinsic semantic features. But again, we will make room for such cases in the formal theory.

The case of S indicates that truth functional components of liar-like sentences can be crucial to the determination of their value. So the determination set of a sentence involving ‘true’ will include not only the sentences to which it makes reference or quantifies over, but also its truth functional components. So the sentence ‘Snow is white’ of *L* will be in the determination set of S, along with S.

6.2. Basic notions

Let *L* be a fragment of English free of the terms 'denotes', 'extension', 'true', ‘false’, 'true of', ‘false of’. For simplicity, we will further assume that *L* contains no context-sensitive terms, and we will also set aside vague terms. We obtain the language £ by adding to *L* the context-sensitive term *t* (either 'denotes' or 'extension' or 'true'/’false’ or 'true of'/false of'). We may think of ₤ as the non-semantical part of English together with the context-sensitive English term *t*. We'll reserve the term 'expression' to pick out *token* expressions that are denoting phrases or predicates or sentences.

Suppose we have an expression σ in context c_σ in which the term *t* occurs - σ is a referring expression token containing 'denotes', or a predicate token containing 'extension', or a sentence token containing 'true' or 'false', or a predicate token containing 'true of' or 'false of'. This occurrence of *t* is represented by '$t_{c\sigma}$'. In general, any token of *t* occurring in context α is represented by 't_α'. And this representation is extended to any token of *t* *co-extensive* with a token of *t* occurring in context α. This allows the introduction of the notion of an *α-schema*: an α-schema is a schema for denotation, extension, truth or truth-of in which the occurrence of *t* on the left-hand-side is represented by 't_α'.

Given an expression σ in which *t* occurs, and given a context κ, we now define the notions of a *representation of* σ, a *primary representation of* σ, and a *secondary representation* of σ.

A *representation* of an expression σ is an ordered triple <type(σ),c_σ,κ>, where

(i) The first element indicates the type of σ.

(ii) The second element indicates that the occurrence of *t* in σ is represented by '$t_{c\sigma}$'.

(iii) The third element indicates that σ is to be evaluated in the context κ.

The second element of a representation is associated with the c_σ-schema (in which the appearance of *t* on the left-hand-side is represented by '$t_{c\sigma}$'). This is the schema by which

σ's determinants are evaluated. The context c_σ fixes the schema that evaluates σ's determinants. The context κ fixes the schema, or the kind of schema, by which σ itself is evaluated.[2] The context κ may fix a schema different from the c_σ-schema, or a kind of schema to which the c_σ-schema does not belong.

We now turn to a preliminary definition of a primary representation:

(Preliminary form) The *primary representation* of σ is a representation where κ is c_σ, σ's context of utterance. That is, the primary representation of σ is <type(σ),c_σ,c_σ>.

The idea is that the schema or kind of schema by which σ is to be evaluated is the one fixed by c_σ, σ's own context of utterance. In ordinary cases, c_σ will fix the c_σ-schema as the schema that evaluates σ. But as we saw in 4.1, repetitions like C* provide exceptional cases. For example, the context of C* fixes the unreflective c_C-schema as the evaluating schema for C*'s determinants, and a reflective r_C-schema as the kind of schema by which C* itself is to be evaluated. The same goes for variants of repetitions, such as C^v from 4.1 ('the number you get when you add up the numbers denoted by the expressions on the board in room 213').

We saw in 4.1 that this preliminary definition of a primary representation can be improved upon in the case of repetitions, once the facts are in about the repetition's context. If σ is a repetition of ρ, then type(σ)=type(ρ), the occurrence of t in σ is represented by '$t_{c\rho}$', and c_σ fixes an r_ρ-schema as the evaluating schema for σ. So the primary representation of a repetition σ is best given as <type(ρ),c_ρ,r_ρ>. And the primary representation of a variant repetition τ of ρ is <type(τ),c_ρ,r_ρ>, since, unlike a repetition of

ρ, a variant repetition is not of the same type as ρ. Repetitions and variant repetitions are accommodated in the following definition of a primary representation.

If σ is not a repetition, its *primary representation* is $<$type(σ),c_σ,$c_\sigma$$>$. If σ is a repetition of ρ, its *primary representation* is $<$type(ρ),c_ρ,$r_\rho$$>$; if σ is a variant repetition of ρ, its *primary representation* is $<$type(σ),c_ρ,$r_\rho$$>$.

A *secondary representation* of σ is any representation of σ that is not a primary representation.

If σ is an expression of *L*, and so contains no occurrence of *t*, *we represent σ by a unit sequence <type(σ)>, and call this its primary representation*.

If an expression σ contains an occurrence of *t*, then the determination of a value for σ will depend on the values of other expressions to which σ makes reference or quantifies over, and also, in the case of truth, the values of expressions which are truth-functional components of liar-like sentences. These expressions are members of the ***determination set*** for σ.

Given the notion of a determination set, we can now define in a preliminary way the notion of a primary tree. Notice that the nodes of the primary tree are representations. Some terminology: *if n is a node, then n_i is the ith member of the ordered triple, for $1 \leq < i \leq 3$.*

(Preliminary form) Given an expression σ containing an occurrence of the term t, the *preliminary primary tree for σ* is given as follows:

(1) The top node of the tree is the primary representation of σ.

(2) For any node *n*=<type(τ),c_τ,α> on the tree that represents an expression τ,

(a) τ has a non-empty determination set S, and

(b) the nodes immediately below *n* are representations of the members of S, and for any such representation *m* that is a triple, $m_3 = c_\tau$.

(3) If a node is of the form <type(λ)>, the branch terminates at that node.

A branch of a primary tree will either be infinite or terminate at an expression of *L*. Some expressions containing *t* will not have a primary tree. Consider for example the phrase 'the sum of the numbers denoted by expressions on the board', uttered in circumstances where nothing is written on the board. In this case, clause 2(a) is not satisfied.

This definition of a preliminary primary tree does not yet build in relevant information about explicitly reflective contexts and schemas. To obtain the primary tree for σ, we can apply the procedure described in 6.1.1.

Given an expression σ containing an occurrence of the term t, the *primary tree* for σ is obtained via the following procedure:

(i) Take the preliminary primary tree for σ. If the second or third element of any triple on the tree indicates a schema that is explicitly reflective, add an additional subscript to that element to denote the set of expressions to which the schema is

explicitly reflective. If no explicitly reflective schemas are indicated, then the primary tree for σ is just the preliminary primary tree.

(ii) Identify any representation of an expression ρ, of the form <type(ρ),cρ,$[c\tau]_U$>, where ρ is a member of the set U. Replace that representation by the triple <type(ρ),cρ,rρ>, and remove the additional subscript 'U' from all nodes.

For definitions of the notions of a ***preliminary secondary tree*** and a ***secondary tree*** for σ, simply replace the term 'primary' by 'secondary' in the previous two definitions.

6.3 The 0-expressions

We first consider those expressions – call them the *0-expressions* - that depend for a value only on expressions of *L*. When we speak of the value of an expression σ, we are speaking of σ's denotation if σ is a referring expression, or its extension if it is a predicate, or its truth value if it is a sentence. An expression σ has a value relative to an evaluating schema, so values attach, if at all, only to *representations* of an expression. For example, P has no extension_{cP} - that is, no value attaches to the representation <type(P),c_P,c_P>. But P does have an extension_{rP} - that is, a value attaches to the representation <type(P),c_P,r_P>. Though values attach, or fail to attach, to *representations*, it is sometimes convenient to speak of *the value of an expression* as a shorthand way of speaking of *the value of its primary representation*. We now work towards a characterization of the 0-expressions.

The *L-tree* for σ is the tree obtained from σ's primary tree by retaining only finite branches.

If *n* is a node on the *L*-tree for σ, then the *L-tree determinants* of *n* are the nodes immediately below n.

Each branch of σ's *L*-tree terminates at a unit sequence that represents an expression of *L*. These terminating nodes will either have a definite value, or they won't. (For example, the predicate 'natural number' will; the referring expression 'the natural number between 6 and 7' will not.) There are three possibilities for a node *n* of σ's *L*-tree:

(i) *n* has a definite value, determined by its *L*-tree determinants.

(ii) it is definite that *n* has no value, given its *L*-tree determinants.

(iii) a value for *n* cannot be determined from its *L*-tree determinants. (Whether or not *n* has a value cannot be determined solely via expressions of *L*.)

For example, consider the case from 6.1 in which I write on the board the following predicates:

(N) moon of the Earth

(Q) extension of a predicate on the board with at least one member.

The top node of the L-tree for Q is a case of (i). Contrast the case where I write:

(N) moon of the Earth.

(R) extension of a predicate on the board with at least two members.

Here the top node of the *L*-tree for R is a case of (ii).

And if I write:

(N) moon of the Earth

(S) unit extension of a predicate on the board,

then the top node of the *L*-tree for S is a case of (iii).

We can now describe in a general way a procedure for pairing values with nodes of the L-tree, for those nodes that have values. Start with the terminal nodes. These nodes represent expressions of *L,* and they will straightforwardly have values (like "Snow is white") or fail to have values (like "the integer between 5 and 6"). The value status of these nodes can be settled one way or the other. Now consider any node *n* all of whose *L*-tree determinants are terminal nodes. Proceed by determining whether (i), (ii) or (iii) holds for *n*, given its *L*-tree determinants. Continue up the tree, associating with each node its value if it has one. Eventually, the primary representation of σ at the top of the tree is reached. The same three possibilities hold for the top node of the tree as for all the other nodes. We will say that σ is a 0-expression if either (i) or (ii) is the case for the top node of its primary tree.

A node *n* of an *L*-tree is *settled* if the procedure just described either determines a value for *n* or determines that *n* has no value.

σ is a *0-expression* iff the top node of its *L*-tree is settled.

Whether or not the primary representation of a 0-expression gets a value depends solely on (the values of) expressions of *L*. The predicates Q and R are 0-expressions; the predicate S is not. Trivially, an expression of *L* is a 0-expression.

6.4 The reflective hierarchy

We can think of the 0-expressions as reflection-free, in the sense that they do not depend for their values on any reflective evaluation of a pathological expression.[3] A 0-expression depends for its value (or lack of value) on the nodes of its *L*-tree, and no reflective evaluation of a pathological expression ρ can appear on an *L*-tree. Our account below will guarantee this, but we can give the intuitive idea now. Recall that an *L*-tree for σ is obtained from σ's primary tree. If a reflective evaluation of expression ρ appeared on a branch of σ's primary tree, then ρ itself will be represented on this branch -- and so the branch will be infinite. But an L-tree has no infinite branches.

There are other expressions that are also reflection-free but not 0-expressions. Clearly, none of the pathological expressions C, P, L, and H are 0-expressions: their values cannot be determined from the values of expressions of the language *L* alone. But these expressions do not involve reflective evaluations in any way: they are not themselves reflective, and the network of sentences displayed by their primary trees contain no reflective expressions. So the values of C, P, L, and H do not depend at all on reflective utterances. Below we will define the *1-expressions* as those that are reflection-free. Among the 1-expressions are all the 0-expressions and the pathological expressions C, P, L, and H.

Since C, P, L and H are all pathological, we may go on to reflect upon them, and produce reflective utterances. For example, consider again the rehabilitation of C, produced in a suitably reflective context:

(†) The number denoted by C is $\pi+6$.

We let D be the first five words of (†). As we saw in 6.1.1, the primary tree for D is:

$$\begin{tikzcd}
 & \langle \mathrm{type}(D), c_D, c_D \rangle \arrow[d, dash] & & \\
 & \langle \mathrm{type}(C), c_C, r_C \rangle \arrow[d, dash] & & \\
 & \langle \mathrm{type}(C), c_C, c_C \rangle \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & & \\
\mathrm{type}(A) & \mathrm{type}(B) & \langle \mathrm{type}(C), c_C, c_C \rangle \arrow[dl, dash] \arrow[d, dash] & \\
 & \mathrm{type}(A) & \mathrm{type}(B) & \ddots
\end{tikzcd}$$

The second node from the top indicates that C is to be assessed by a reflective schema. Both C* and C† occur in contexts that are reflective with respect to C; below we shall characterize C* and C† as *1-reflective*, since they are produced in contexts that are reflective with respect to a 1-expression. And we will say that the 1-reflective expressions comprise the *2-expressions*.

Now notice that we can continue on from our production of C†, perversely adding: "And so the number denoted by C, plus the number denoted by '2^2', plus the sum of the numbers denoted by phrases in this utterance, is irrational." Let C‡ be the final definite description token in our utterance (the token that begins with 'the sum of'). C‡ is pathological. Now since C‡ is produced in a context that is reflective with respect to a 1-expression, it is *1-reflective*, and so a 2-expression. So there are pathological 1-reflective expressions.

We can in turn reflect on C‡, and produce expressions in a context reflective with respect to C‡. We will characterize these as 2-reflective, since they are reflective with respect to pathological 2-expressions. And these 2-reflective expressions will comprise the 3-expressions. And so on through the levels.

We are led in this way to a hierarchy of expressions. Call this hierarchy *the* *reflective hierarchy*. The level of an expression σ in the hierarchy is *a measure of the highest reflection that σ involves*; for example, $C^{\ddagger}$ is a 2-expression, and the highest reflective evaluation that it involves is a reflective evaluation of a 1-expression. *The level is not a measure of the scope of an occurrence of the term t in σ* (whether *t* is 'denotes', 'extension', 'true' or 'true of'). For example, $C^{\dagger}$ occupies a higher level of the hierarchy than C. But as we have seen in 3.3, the extension of 'denotes' in C is neither narrower nor broader than the extension of 'denotes' in $C^{\dagger}$ - neither extension includes the other, and each has singularities that the other does not. *We are not offering a Tarskian treatment of t*. We are not stratifying the term *t* into distinct levels, where the higher the level the broader the scope. We reject such a hierarchical treatment of the term *t*. Instead, we retain a single context-sensitive term, whose occurrences all have singularities, and where no occurrence is more comprehensive in its scope than another. Notice another anti-hierarchical feature of the singularity account: any occurrence of *t* will contain within its scope expressions from all levels of the reflective hierarchy, since at every level there will be expressions that are not singularities of *t* (for example, any non-pathological expressions of any level).

We need the reflective hierarchy in order to realize the main aim of the formal theory: the identification of singularities of occurrences of t in an expression σ. The singularities of t in an expression σ will depend on the semantical status of σ. It matters whether σ is a 0-expression, or pathological, or reflective. And we cannot properly determine the status of σ until we have identified its position in the reflective hierarchy.

We now take the first of a number of steps toward a more precise characterization of the reflective hierarchy and singularities.

6.5 Reflection-free expressions and singularities

At the lowest level of the reflective hierarchy are the reflection-free expressions, those expressions that do not depend for their value on reflective evaluations. We want to characterize these expressions in a more precise way. And we want to characterize the pathological reflection-free expressions and identify their singularities. To these ends, we begin by defining the notion of a *pruned$_0$ tree*.

Given an expression σ, the *pruned$_0$ tree* for σ is obtained from σ's primary tree by terminating any branch at the first occurrence of a primary representation of a 0-expression.

Once we arrive at the primary representation of a 0-expression, we arrive at a node for which we can supply a value (via the values of *L*-expressions alone). And so we terminate the branch there.

An infinite branch on a pruned$_0$ tree indicates pathology. There are two kinds of pathology: loops and chains. Turning first to loops, a pathological expression σ may include itself in its own determination set – in this sense, it is self-referential. In this case, σ's pruned$_0$ tree will have an infinite branch on which σ's primary representation repeats (for example, consider the cases of C and P). But loops can also be wider – as in the case of Fran and Grace from 3.3. Fran produces the expression "the sum of one, two, and the number denoted by Grace's current utterance", and Grace is saying "the sum of one, two and the number denoted by Fran's current utterance". The pruned$_0$ tree (which is just the primary tree) for Fran's utterance F can be presented this way:

```
        <type(F),c_F,c_F>
           /      |        \
<type(1)>   <type(2)>   <type(G),c_G,c_F>
                          /       |          \
                  <type(1)>   <type(2)>   <type(F),c_F,c_G>
                                            /        |         \
                                    <type(1)>   <type(2)>   <type(G),c_G,c_F>
                                                              /        |          \
                                                      <type(1)>   <type(2)>  <type(F),c_F,c_G>
                                                                                /     |        .
                                                                        type(1)>  <type(2)>  .
                                                                                                .
```

As we noted in Chapter 4, the distinctive feature of this infinite branch of F's pruned$_0$ tree is that every secondary representation repeats, including the secondary representation of F. And by Symmetry, since G cannot be assessed by the c_F-schema, neither can F. By Symmetry, then, F is pathological, since it cannot be assessed by its associated schema.

Given an expression σ, and an infinite branch of σ's **pruned$_0$ tree, the branch is a *loop* if the primary representation of σ or a secondary representation of σ repeats on the branch.**

Turning now to chains: an expression σ may make reference to an expression ρ, which in turn makes reference to τ, and so on, *ad infinitum* and without repetition. Here, σ heads a chain and this can be captured as follows.

Given an expression σ, and an infinite branch of σ's **pruned$_0$ tree, the branch is a *chain* if no node on the branch repeats.**

Not every expression that heads a chain is pathological – consider, for example, consider the explicitly reflective expression (0^+) from 6.1.1.

We are now in a position to characterize the reflection-free expressions.

σ is a *reflection-free* expression iff

(a) σ is a 0-expression, or

(b) (i) every infinite branch of σ's pruned$_0$ tree (if there are any) is a loop or a chain.

(ii) no node of σ's pruned$_0$ tree is of the form $<$type(ρ),c_ρ,$r_\rho$$>$.

As we have seen, a reflection-free expression is one whose value does not depend on any reflections. Clause (b)(i) ensures that σ's value does not depend on any non-explicit reflective evaluation. If σ is caught in a loop, then it is a member of a symmetric network, in which no member stands above the others. And by Symmetry no non-explicit reflective evaluations are associated with chains. Clause (b)(ii) ensures that σ's value does not depend on any explicitly reflective evaluation. We can now define the 1-expressions:

σ is a *1-expression* iff σ is reflection-free.

The 1-expressions form the lowest level of the reflective hierarchy. Next we turn to the pathological reflection-free expressions:

A reflection-free expression σ is *pathological* if the pruned$_0$ tree for σ is unfounded.

We can now state precisely the principle of Symmetry which underlies this characterization of pathologicality for reflection-free expressions.

Let σ be a pathological reflection-free expression. The expressions represented by nodes of an infinite branch of σ's pruned$_0$ tree form a *reflection-free symmetrical network*.

Principle of Symmetry **All members of a reflection-free symmetrical network share the same pathological status.**

We now turn to a crucial class of singularities associated with the reflection-free expressions: the key singularities.

Consider a pathological reflection-free expression σ. Take any node n of an infinite branch of σ's pruned$_0$ tree, and let ρ be the expression for which n is a representation (primary or secondary). We will have $n_3=c_\upsilon$, for some expression υ. Then ρ is a *key singularity* of $t_{c\upsilon}$.

When we attempt to determine a value for ρ by the c_υ-schema, we fail - this failure is indicated by the infinite branch.

If n is not the top node, then, since, $n_3=c_\upsilon$, there is a node m immediately above n such that $m_2 = c_\upsilon$. In order to determine a value for m, it will be crucial to exclude the key singularity ρ from the scope of ‘$t_{c\upsilon}$’. For example, it is easy to check that C is a key singularity of 'denotes$_{cC}$'. And once C is excluded from the extension of the occurrence of 'denotes' in C, we can determine a value for C (namely, π+6). In general, the removal of the key singularities of '$t_{c\upsilon}$' from its extension is a key step towards establishing a value for nodes whose second member is c_υ.

For another example, consider again (0) (introduced in 6.1). The pruned$_0$ tree for (0) is identical to its primary tree:

$$\begin{tikzcd}
<\text{type}(0),c_0,c_0> \arrow[d, dash] \\
<\text{type}(1),c_1,c_0> \arrow[d, dash] \\
<\text{type}(2),c_2,c_1> \arrow[d, dash] \\
<\text{type}(3),c_3,c_2> \\
\vdots
\end{tikzcd}$$

It is straightforward to check that the key singularity of 'true$_{c0}$' is (1), the key singularity of 'true$_{c1}$' is (2) – and in general, the key singularity of 'true$_{cn}$' is (n+1). Once we remove (1) from the extension of 'true$_{c0}$', a value can be determined for (0) (namely, false – which corresponds to its reflectively established value). And the value of all the other sentences in the chain can be determined once the respective key singularities are excluded (false for the even-numbered sentences, true for the odd-numbered).

Notice that (0), (1), (2), ... form a symmetrical network, and so, by Symmetry, (0), (1), (2), … are all singularities of the occurrences of 'true' in (0), (1), (2), ... In general, suppose that σ is a pathological reflection-free expression. Consider an infinite branch of its primary tree, and its associated symmetric network. Then, given an occurrence of *t* in a member of the symmetrical network, it is a consequence of Symmetry that every member of the symmetrical network is a singularity of that occurrence of *t*.

The pruned$_0$ tree for a pathological reflection-free expression σ frames a procedure for assigning a value to (the primary representation of) σ. For this reason, we will call the pruned$_0$ tree for σ its ***determination tree***. There are two kinds of branches on a determination tree. One kind terminates at a primary representation of a 0-expression. The other kind is an infinite branch. The second node of an infinite branch is of the form $<\text{type}(\rho),c_\rho,c_\sigma>$, where c_σ is the second element in the primary representation of σ. And this node will represent a key singularity of the occurrence of '$t_{c\sigma}$' in σ. The value of σ (if it has one) will be determined via the values of certain 0-expressions (those whose primary representations are terminal nodes), and the exclusion of the key singularities from the extension of '$t_{c\sigma}$'. The relevant 0-expressions and the key singularities can be read off σ's determination tree.

For example, consider again the pruned$_0$ tree for C:

$$\begin{tikzcd}
 & <\text{type}(C),c_C,c_C> \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & & \\
<\text{type}(A)> & <\text{type}(B)> & <\text{type}(C),c_C,c_C> \arrow[dl, dash] \arrow[d, dash] & \\
 & <\text{type}(A)> & <\text{type}(B)> & \ddots
\end{tikzcd}$$

The second node of the infinite branch indicates that C is a singularity of the occurrence of ‘denotes’ in C. And the tree as a whole indicates that the value for C depends upon the values of the 0-expressions A and B, and the fact that C is excluded from the extension of 'denotes$_{cC}$'. The tree presents the elements of a reflective evaluation of C: we can reflectively assign the value π+6 to C because A denotes$_{cC}$ π, B denotes$_{cC}$ 6, and C does not denote$_{cC}$ at all.

For another example, consider the pruned$_0$ tree for the Liar sentence L:

$$\begin{tikzcd}
<(L),c_L,c_L> \arrow[d, dash] \\
<(L),c_L,c_L> \arrow[d, dash] \\
\vdots
\end{tikzcd}$$

This tree indicates that a value for L depends solely on the fact that L is a singularity of 'true$_{cL}$'. This is the fact that is used to determine reflectively the truth value *true* for L, since L says it isn't true$_{cL}$, and indeed it isn't, because it is a singularity of 'true$_{cL}$'.

In general, every 1-expression σ has a determination tree, via which a value for σ can be established, if it has one. A value for σ may not be forthcoming from the information supplied by the determination tree: there are 1-expressions that have no value, just as there are 0-expressions that have no value.   For example, consider this chain of denoting phrases: (1) the number denoted by (2); (2) the number denoted by (3); … ; (n) the number denoted by (n+1); … .   The phrase (1) heads a chain. Its pruned$_0$ tree is:

$$\begin{tikzcd}\langle \mathrm{type}(1),c_1,c_1\rangle \arrow[d, dash] \\ \langle \mathrm{type}(2),c_2,c_1\rangle \\ \vdots \end{tikzcd}$$

This determination tree indicates that the only information we have for determining a value for (1) is that (2) is a singularity of 'denotes$_{c1}$'. So (2) is excluded from the extension of 'denotes$_{c1}$'. But since (2) is the only member of (1)’s determination set, no value can be assigned to (1).

This completes our description of the lowest level of the reflective hierarchy, the level occupied by the 1-expressions. These are exactly the reflection-free expressions, the expressions whose values do not depend on reflective evaluations.

6.6 Higher levels of the reflective hierarchy

We now turn to the higher levels of the reflective hierarchy. The first level of the hierarchy is occupied by the 1-expressions. The next level – the 2-expressions – are those that involve reflections on pathological 1-expressions, and so on. To characterize the higher levels of the reflective hierarchy, we proceed by (complete) transfinite induction.

Given the δ-expressions for $1 \leq \delta < \alpha$, we will show how to characterize the α-expressions. There are two cases to consider: where α is a successor ordinal, and where α is a limit ordinal.

First, then, we consider the case where α is a successor ordinal β+1. Given the δ-expressions for $1 \leq \delta \leq \beta$, we can go on to characterize the β+1-expressions via the notions of a *reflective δ-representation*, a *pruned$_\beta$ tree,* and a *β-reflective expression*.

For δ ≥1, a representation *n* of a δ-expression τ is a *reflective δ-representation* if *n* is a secondary representation of τ where either $n_3=r_\tau$ or n is the only node not to repeat on an infinite branch of the associated secondary tree for τ (i.e. the secondary tree for τ with top node *n*).[4]

A reflective δ-representation provides an explicit or non-explicit reflective evaluation of a (pathological) δ-expression.

Given an expression σ and an ordinal β, the *pruned$_\beta$ tree for σ* is obtained from σ's primary tree by terminating any infinite branch at the first occurrence of a reflective δ-representation, for all δ such that 1≤δ≤β.

Given an expression σ, and an infinite branch of σ's **pruned$_\beta$ tree, the branch is a *loop* if the primary representation of σ or a secondary representation of σ repeats on the branch.**

Given an expression σ, and an infinite branch of σ's **pruned$_\beta$ tree, the branch is a *chain* if no node on the branch repeats.**

For ordinal β, an expression σ is *β-reflective* iff the following is true of σ's pruned$_\beta$ tree:

(i) one of its nodes is a reflective β-representation.

(ii) every infinite branch (if there are any) is a loop or a chain.

(iii) no node of the form <type(ρ),c_ρ,r_ρ> appears on an infinite branch.

A β-reflective expression involves at least one reflective evaluation of a β-expression. But clauses (ii) and (iii) guarantee that it does not involve any higher reflective evaluations. Now we can characterize the β+1-expressions:

Given an ordinal β≥0, the *β+1-expressions* are

(i) the reflection-free expressions, if β=0

(ii) the β-reflective expressions, if β≠0.

For example, the repetition C* is a 2-expression since it is a reflective 1-representation. The pruned$_1$ tree for C* is simply

<type(C),c_C,r_C>

since the primary tree for C* is terminated at <type(C),c_C,r_C>, the first reflective 1-representation that appears on an infinite branch. This tree satisfies clause (i) of the definition of a β-reflective expression for β=1, and clauses (ii) and (iii) are vacuously satisfied, since there are no infinite branches on the tree.

For another example, consider again Fran and Grace’s looped utterances. (Fran: "the sum of one, two, and the number denoted by Grace's current utterance"; Grace: "the sum of one, two and the number denoted by Fran's current utterance".) Suppose a third party Hugo produces the expression:

(H) The number denoted by Fran's utterance.

in a neutral context. Then it's easy to see that H is a 1-reflective expression. The primary tree for H looks like this:

```
          <type(H),c_H,c_H>
                 |
          <type(F),c_F,c_H>
          /      |        \
<type(1)>   <type(2)>   <type(G),c_G,c_F>
                        /       |         \
               <type(1)>   <type(2)>   <type(F),c_F,c_G>
                                       /        |        \
                              <type(1)>  <type(2)>   <type(G),c_G,c_F>
                                                     /        |         \
                                            <type(1)>   <type(2)>  <type(L),c_F,c_G>
                                                                   /       |        .
                                                           type(1)>  <type(2)>    .
                                                                                    .
```

To check that the node <type(F),c_F,c_H> is a reflective representation of the 1-expression F, observe that the representation <type(F),c_F,c_H> is a secondary representation of F, and the associated secondary tree for F is:

```
        <type(F),c_F,c_H>
          /      |        \
<type(1)>   <type(2)>   <type(G),c_G,c_F>
                        /       |         \
               <type(1)>   <type(2)>   <type(F),c_F,c_G>
                                       /        |        \
                              <type(1)>  <type(2)>   <type(G),c_G,c_F>
                                                     /        |         \
                                            <type(1)>   <type(2)>  <type(F),c_F,c_G>
                                                                   /       |        .
                                                           type(1)>  <type(2)>    .
                                                                                    .
```

The node <type(F),c_F,c_H> is the first node not to repeat on the infinite branch of this secondary tree for F.   So <type(F),c_F,c_H> is a reflective 1-representation of F, and so the pruned$_1$ tree for H is:

$$\begin{tikzcd}
\langle \mathrm{type}(H),c_H,c_H\rangle \arrow[d, dash] \\
\langle \mathrm{type}(F),c_F,c_H\rangle
\end{tikzcd}$$

It follows that H is 1-reflective, and so it is a 2-expression.

For another example, consider again the pathological expression C‡ from 6.4 above. C‡ is the final definite description in the following continuation following our production of C†:

> And so the number denoted by C, plus the number denoted by '2^2', plus the sum of the numbers denoted by phrases in this utterance, is irrational.

The pruned$_1$ tree for C‡ is:

$$\begin{tikzcd}
 & \langle \mathrm{type}(C^{\ddagger}), c_{C^{\ddagger}},c_{C^{\ddagger}}\rangle \arrow[dl, dash] \arrow[d, dash] \arrow[drr, dash] & & & \\
\langle \mathrm{type}(C^{\dagger}),c_{C^{\dagger}},c_{C^{\ddagger}}\rangle \arrow[d, dash] & \langle \mathrm{type}(2^2)\rangle & & \langle \mathrm{type}(C^{\ddagger}),c_{C^{\ddagger}},c_{C^{\ddagger}}\rangle \arrow[dl, dash] \arrow[d, dash] & \\
\langle \mathrm{type}(C),c_C,r_C\rangle & & \langle \mathrm{type}(C^{\dagger}),c_{C^{\dagger}},c_{C^{\ddagger}}\rangle \arrow[d, dash] & \langle \mathrm{type}(2^2)\rangle & \ddots \\
 & & \langle \mathrm{type}(C),c_C,r_C\rangle & &
\end{tikzcd}$$

The procedure that takes us from C‡'s preliminary primary tree to its primary tree yields the nodes <type(C),c_C,r_C>, since the $c_{C^{\dagger}}$-schema is reflective with respect to C. The branches of the primary tree on which thsse nodes appear are infinite. We obtain the pruned$_1$ tree by terminating those branches at those nodes. So the finite branches of the pruned$_1$ tree terminate either at the (primary) representations of the 0-expression '2^2', or the secondary representation <type(C),c_C,r_C> of the 1-expression C. (It is easy to check that the representation <type(C),c_C,r_C> is the only node not to repeat on the associated secondary tree for C.) The rightmost branch of the pruned$_1$ tree for C‡ does not terminate, indicating the pathological character of C‡. This pruned$_1$ tree satisfies clauses (i)-(iii) in the definition of a β-reflective expression for β=1, and so C‡ is a 1-reflective expression, and a 2-expression.

We now move on to the case of the α-expressions, where α is a limit ordinal. To motivate the formal treatment, consider the following case (which we will consider in full detail in the next chapter). We have the following network of denoting expressions, where a_n, b_n and c_n are the only expressions written on the board in room n, and where the token d is written elsewhere.

(d) the sum of the numbers denoted by c_0, c_1, c_2, ... c_n, … .

(a_0) 0
(b_0) the square of 0.
(c_0) the sum of the numbers denoted by expressions in room 0.

(a_1) 0
(b_1) the square of 0.
(c_1) the sum of the numbers denoted by c_0 and the expressions in room 1.
.
.
.

(a_{n+1}) 0
(b_{n+1}) the square of 0.
(c_{n+1}) the sum of the numbers denoted by c_n and the expressions in room n+1.
.
.
.

It is straightforward to check that c_0 is a reflection-free expression, c_1 is 1-reflective, c_2 is 2-reflective, and in general, c_n is n-reflective. Now consider d, and the

pruned_n tree for d for any n. The top node – the primary representation of d - does not repeat on any infinite branch of the pruned_n tree, or head an infinite chain. So d is not n-reflective, for any finite ordinal n. But intuitively, d is reflective with respect to c_0, c_1, c_2, ... So we need to accommodate transfinite levels – in particular, we need to accommodate the α-expressions, for α a limit ordinal.[5] We now work towards a definition of these α-expressions.

Given an expression σ and a limit ordinal α, the *$\text{pruned}_{<\alpha}$ tree for σ* is obtained from σ's primary tree by terminating any infinite branch at the first occurrence of a reflective β-representation, for all β such that $1 \leq \beta < \alpha$.

For α a limit ordinal, σ is *<α-reflective* iff

(i) σ's primary tree contains a reflective β-representation, where $\beta \geq 1$ and α is the least limit ordinal such that $\beta < \alpha$.

(ii) for each reflective β-representation on σ's primary tree, there is a reflective γ-representation on σ's primary tree, where $1 \leq \beta < \gamma < \alpha$.

(iii) the following is true of the *$\text{pruned}_{<\alpha}$ tree for σ*:

> **(a) every infinite branch (if there are any) is a loop or a chain.**
>
> **(b) no node of the form $<\text{type}(\rho), c_\rho, r_\rho>$ appears on an infinite branch.**

For a limit ordinal α, *the α-expressions* are the <α-reflective expressions.

We are now is a position to define the α-expressions for *any* ordinal $\alpha \geq 1$.

Given $\alpha \geq 1$, the *α-expressions* are

(i) the reflection-free expressions if $\alpha=1$

(ii) the β-reflective expressions if α is a successor ordinal $\beta+1$

(iii) the $<\alpha$-reflective expressions if α is a limit ordinal.

It is worth noting that the notion of an α-expression is not cumulative: the class of α-expressions does not contain expressions from lower levels. It is also worth noting again that the level of an expression σ in the reflective hierarchy is not a measure of the extension of the term *t* occurring in σ. It is a measure only of the level of the highest reflection that is associated with σ.

We now capture the notion of pathology for an α-expression.

An α-expression is *pathological* iff

(i) α is a successor ordinal $\beta+1$ ($\beta \geq 0$), and the pruned$_\beta$ tree for σ is unfounded,

or (ii) α is a limit ordinal, and the pruned$_{<\alpha}$ tree for σ is unfounded.

We can now state precisely the principle of Symmetry which underlies this characterization of pathologicality for α-expressions.

Let σ be a pathological α-expression. The expressions represented by nodes of an infinite branch of σ's pruned$_\beta$ tree, for α a successor ordinal $\beta+1$, or σ's pruned$_{<\alpha}$

tree, for α a limit ordinal, form a *symmetrical network*.

Principle of Symmetry **All members of a symmetrical network share the same pathological status.**

Now we characterize the key singularities in a general way, for a pathological expression σ, and the occurrence of '$t_{c\sigma}$' in σ.

Let σ be a pathological α-expression. Let n be any node of an infinite branch of σ's pruned$_{\beta}$ tree (for α a successor ordinal β+1) or σ's pruned$_{<\alpha}$ tree (for α a limit ordinal), and let n be a representation of ρ. Suppose $n_3 = c_\sigma$. Then ρ is a *key singularity* of '$t_{c\sigma}$'.

Recall the pruned$_1$ tree for the 2-expression C‡:

$$\begin{tikzcd}[column sep=small]
 & & \langle type(C^{‡}),c_{C‡},c_{C‡}\rangle \arrow[dll, dash] \arrow[d, dash] \arrow[drr, dash] & & & & \\
\langle type(C^{†}),c_{C†},c_{C‡}\rangle \arrow[d, dash] & & \langle type(2^2)\rangle & & \langle type(C^{‡}),c_{C‡},c_{C‡}\rangle \arrow[dl, dash] \arrow[dr, dash] & & \\
\langle type(C),c_C,r_C\rangle & & & \langle type(C^{†}),c_{C†},c_{C‡}\rangle \arrow[d, dash] & & \langle type(2^2)\rangle & \vdots \\
 & & & \langle type(C),c_C,r_C\rangle & & &
\end{tikzcd}$$

The rightmost branch is infinite, and it's easy to see that C‡ is a key singularity of 'denotes$_{cC‡}$'.

With the notion of a singularity in hand, we can see how *a pruned tree frames a procedure for determining a value for an α-expression*. A pruned tree has at most three kinds of branches: those that are infinite, those that terminate in a reflective representation, and those that terminate in a (primary) representation of a 0-expression. For example, the pruned$_1$ tree for $C^{\ddagger}$ has an infinite branch, branches that terminate at the reflective representation $<type(C),c_C,r_C>$, and branches that terminate at the primary representation of the 0-expression '2^2'. Each kind of branch contributes to the determination of a value for $C^{\ddagger}$.

In particular, the second node of the infinite branch indicates a key singularity of the occurrence of 'denotes' in $C^{\ddagger}$. The second node represents an expression in $C^{\ddagger}$'s determination set that cannot be evaluated by the $c_{C\ddagger}$-schema. The key singularity is $C^{\ddagger}$ itself, since $C^{\ddagger}$ belongs to its own determination set. So when we determine a value for the phrase $C^{\ddagger}$ via the members of its determination set, we exclude $C^{\ddagger}$, and consider only the values of the other phrases in $C^{\ddagger}$'s determination set.

In determining a value for $C^{\ddagger}$, then, we consider only the finite branches of its pruned$_1$ tree. One type of finite branch terminates in the representation of a 0-expression '2^2' whose value is clear. The other type of finite branch terminates in the reflective representation $<type(C),c_C,r_C>$, which is associated with a reflective evaluation of C. And, as we noted in 6.5, a procedure for determining a reflectively established value for C is in turn framed by the pruned$_0$ tree for C:

$$\begin{tikzcd}[arrows=dash]
 & <type(C),c_C,c_C> \arrow[dl] \arrow[d] \arrow[dr] & & \\
<type(A)> & <type(B)> & <type(C),c_C,c_C> \arrow[dl] \arrow[d] & \\
 & <type(A)> & <type(B)> & \ddots
\end{tikzcd}$$

This tree presents the elements of a reflective evaluation of C: we can reflectively assign the value $\pi+6$ to C because A denotes$_{cC}$ π, B denotes$_{cC}$ 6, and C is a key singularity of ‘denotes’ in C and does not ‘denote$_{cC}$’ at all. As with the pruned$_1$ tree for C‡, we attend only to the finite branches of C’s pruned$_0$ tree.

So the terminal nodes of C‡’s pruned$_1$ tree are each associated with definite values, and we can move up through the finite branches of the tree, and determine a value for C‡. It’s easy to check that we obtain the value π +10 for C‡.

We’ll call C‡’s pruned$_1$ tree its *determination tree*, because the tree frames a procedure for determining C‡’s value.

<u>Definition</u>   Given an α-expression σ (α≥1), its *determination tree* is σ's pruned$_\beta$ tree for α a successor ordinal β+1, or σ's pruned$_{<\alpha}$ tree for α a limit ordinal.

For another example, recall the 2-expression H (“The number denoted by Fran’s utterance”), which stands above the Fran-Grace loop. The determination tree for H is H’s pruned$_1$ tree:

$$\begin{tikzcd}
\langle \mathrm{type}(H), c_H, c_H \rangle \arrow[d, dash] \\
\langle \mathrm{type}(F), c_F, c_H \rangle
\end{tikzcd}$$

The node <type(F),c_F,c_H> is associated with a reflective evaluation of Fran’s utterance F. And a procedure for determining this value is in turn framed by the pruned$_0$ tree for (F) (the determination tree for F):

```
       <type(F),c_F,c_F>
        /      |        \
<type(2)>   <type(3)>   <type(G),c_G,c_F>
                          /       |         \
                 <type(2)>   <type(3)>   <type(F),c_F,c_G>
                                           /       |        \
                               <type(2)>  <type(3)>   <type(G),c_G,c_F>
                                                        /         |         \
                                           <type(2)>   <type(3)>  <type(F),c_F,c_G>
                                                                     /       |       .
                                                          type(2)>  <type(3)>    .
                                                                                   .
```

The second node of the infinite branch indicates that G is a key singularity of ‘denotes’ in F, and fails to denote$_F$. So although G is a member of F’s determination set, it is excluded when we determine a value for G – we attend only to the expressions ‘two’ and ‘three’ and determine the value 5 for F. This is the reflectively established value of F which is associated with the node <type(F),c_F,c_H> of H’s pruned$_1$ tree. And this in turn yields the value 5 for H.

In general, the determination tree for an α-expression σ frames a procedure for determining σ’s value. The second nodes of the infinite branches indicate the key singularities of the occurrence of *t* in σ. And these key singularities, though they are members of σ’s determination set, are to be excluded from *t* in σ -- these exclusions contribute to the determination of a value for σ, as do the other members of σ’s determination set. The finite branches of the determination tree end either in a representation of a 0-expression – in which case there is a straightforward associated value for the end node – or in a reflective representation of a pathological expression ρ. The value of this reflective representation is in turn mapped out by the determination tree for ρ. This tree also has up to three kinds of branches, and may include branches that terminate in reflective representations. And so the process continues – but eventually yields representations of reflection-free expressions.

6.7 Summary

We have now completed our description of the reflective hierarchy. At the lowest level are the reflection-free expressions, expressions that do not depend for a value on any reflections. Some of these (the 0-expressions) depend for a value only on the expressions of *L*: the top node of their *L*-trees are settled. Others are pathological, and these further depend for a value on the identification of their key singularities. Their determination trees (the pruned$_0$ trees) frame a procedure for determining their values. As we have seen, we may reflect on pathological expressions, producing expressions at the next level of the hierarchy. And we can go on to produce new pathological expressions that involve these reflections, and in turn reflect upon them. The reflective hierarchy gives formal expression to our ability to reflect upon pathology, produce new pathologies, and reflect in turn upon these – and so on. And the hierarchy also helps us to keep track of the status of an expression. If we can answer questions like: What position does the expression occupy in a network of expressions? Is it pathological? Is it reflective with respect to other expressions, or does it share in their pathology? -- then we can identify the key singularities of a given expression. Once the key singularities are identified, we can remove them from the scope of the given occurrence of *t*. And the removal of the key singularities defuses paradox, and prompts the search for the expression's value. As we saw, the determination tree maps out the procedure for determining a value for σ (if a value exists).

The main task of the formal theory is to identify the key singularities of a given occurrence of 'denotes' or 'extension', or 'true' or 'true of' in an expression of ₤. This we have done by identifying the key singularities of an α-expression, $\alpha \geq 1$. Every expression of ₤ occupies some level of the reflective hierarchy, since there is an upper limit on the level of the highest reflection that a given expression of ₤ involves. So in identifying the key singularities of an α-expression for any ordinal $\alpha \geq 1$, we have identified the key singularities of every expression of ₤.

It's worth noting here that the conceptual resources required to develop the singularity theory are surprisingly meager. For any expression of ₤, we can lay out the primary and secondary trees and identify key singularities if we have just the following: (1) the notion of the type of an expression of ₤, (2) the notion of an expression's context of utterance, (3) contextual input bearing on explicit reflective status, and (4) the determination set of the expression. The conceptual economy of the singularity theory will be relevant in Chapter 9 when we discuss revenge.

It is also worth stressing – as we have before -- that we are not offering a 'Tarskian' solution to the paradoxes. Consider again the case of C and $C^{\ddagger}$. C is a (pathological) reflection-free expression, and $C^{\ddagger}$ is a (pathological) 1-reflective expression. $C^{\ddagger}$ is higher in the reflective hierarchy than C. But it does not follow that the occurrence of 'denotes' in $C^{\ddagger}$ is more comprehensive in scope than the occurrence of denotes in C. Each has singularities that the other does not; neither is broader in scope than the other. In general, consider an occurrence of *t* in a β-expression and an occurrence of *t* in a δ-expression, where $\beta \neq \delta$. The scope of one of these occurrences of *t* is neither narrower nor broader than the other. There is no stratification of the expression *t* into a hierarchy of increasingly comprehensive terms. According to the singularity theory, *t* is a single, context-sensitive term that is minimally restricted on any occasion of use, applying everywhere except to its singularities.

Notes to Chapter 6

---

1. Parallel chains can be constructed for the other semantic notions. Here's a case involving denotation. Suppose there are exactly three denoting expressions on the board: "π", "six" and
(1) the number denoted by (2).

And suppose we introduce the infinite chain of expressions
(2) the number denoted by (3).
(3) the number denoted by (4).
.
.
.
(n) the number denoted by (n+1)
.
.
.

along with the further expression
(0) the sum of the numbers denoted by expressions on the board.

Now consider the following stretch of strengthened reasoning:

> (1) heads an infinite chain, so it is pathological and does not denote a number. So the sum of the numbers denoted by expressions on the board is π+6.

Let (0^+) be the token of the same type as (0) that occurs in this reasoning. By Symmetry, (0) will be treated as a pathological member of an enlarged chain. But (0*) is not pathological, and refers to π+6. Again, the difference between (0) and (0^+) can only be captured in terms of a difference between the contexts of (0) and (0^+) – the latter is reflective with respect to (1), but the former is not.

2. Recall from 4.1 that the context of C* determines only the *kind* of schema that is to evaluate C* -- a schema that is reflective with respect to **C.** The context of C* does not determine a specific schema. At the fourth stage of *repetition*, we produce E, an explicit reflective evaluation of C* -- and here the context determines a *specific* evaluating schema for C*, the reflective c_E-schema.

3. Notice that a representation of a reflective evaluation of a pathological expression might appear on σ's primary tree even though σ is a 0-expression. But this representation will not appear on σ's L-tree.

4. In 7.3.1, we will see a case where a context is explicitly reflective with respect to a whole class of expressions, not just a single expression. One can easily extend the notion of a reflective δ-representation to accommodate this kind of case. Let r_{Cl} indicate a

schema that is explicitly reflective with respect to each member of a class Cl of expressions. Then a secondary representation n of a δ-expression τ is a reflective δ-representation of τ if $n_3=r_{Cl}$ and τ is a member of Cl. For the sake of simplicity, we set this extension aside for now.

5. The example is a variant of one presented in Hardy 1997. Hardy's example establishes the need to accommodate limit ordinals in the case of truth.