Aritmetik Dili

Aritmetik dili, tek birey sabiti '0' olan, fonksiyon işaretleri 's', '+', '·', ve '' olan, yüklemleri ise '<' ve '=' olan dildir. 'ℕ'[1] olarak adlandırdığımız standart modelde, '0' 0 sayısını, adı geçen dört fonksiyon da sırasıyla ardıl almaya, toplamaya, çarpmaya ve üs almaya işaret eder. '<' 'küçüktür'ü, '=' ise eşitliği gösterir.

Daha açık olarak:

- '0' aritmetik dilinin bir terimidir.
- 'x_0', 'x_1', 'x_2', 'x_3'… değişkenleri aritmetik dilinin terimleridir.[2]
- τ ve ρ aritmetik diline ait sembollerse, şunlar da öyledir: sτ, (τ + ρ), (τ·ρ), ve (τ E ρ).
- Bu üç önermede belirtilenlere uygun olmayan hiçbir şey aritmetik dilinin terimi olamaz.

Elimizdeki özelliklerden biri *benzersiz okunabilirlik*tir: her terim benzersiz bir biçimde oluşturulur. Daha açıkça ifade edersek: σ gibi her bir terim için o terimin 'yapı ağacı' olarak adlandırılan, sonlu ve eşsiz bir *doldurulmuş* ağaç vardır. Bu ağacın özellikleri aşağıdaki gibidir:

- Ağacın gövdesi σ ile doludur.
- Ağacın herhangi bir boğumu sτ ile doluysa, o boğumun hemen altında τ ile doldurulmuş bir boğum yer alır.
- Ağacın herhangi bir boğumu (τ + ρ) ile, (τ•ρ) ile, ya da (τ E ρ) ile doluysa, o boğumun hemen altında iki boğum yer alır ki sağdaki ρ, soldakiyse τ ile doludur.
- Ağacın her bir yaprağı (yani uç noktası) ya '0' ile ya da bir değişkenle doludur.

Kapalı terim, hiçbir değişken içermeyen terimdir. Standart modelde, her bir kapalı terim bir doğal sayıyı imler, ve bu şu yolla belirlenir:

- Den ('0') = 0.
- Den (sτ) =Den (τ) + 1.
- Den ((τ + ρ)) = Den (τ) + Den (ρ).

---

[1] Aynı işareti doğal sayılar kümesini göstermek için de kullanıyoruz. Umarım bu bir karışıklığa yol açmaz.

[2] Bu tür değişkenleri yalnızca önemli durumlarda kullanacağız. Çoğu zaman alfabenin sonlarında yer alan öteki harfleri, sadeliği korumak için, alt numaralandırma yapmadan kullanacağız.

- Den $((\tau \cdot \rho)) = \text{Den}(\tau) \cdot \text{Den}(\rho)$.
- Den $((\tau \; E \; \rho)) = \text{Den}(\tau)^{\text{Den}(\rho)}$.

Her bir sayı için bir *standart rakam* tanımlayabiliriz: [n], '0'ın önüne n sayıda 's' yazılarak elde edilen semboldür. O halde, örneğin, [7] = 'sssssss0' (yani 'sssssss0' sayısı için standart rakam '7'dir).

τ simgesini Den (τ) sayısına götüren fonksiyon *hesaplanabilirdir*. Bunun için kullanabileceğimiz bir algoritma şu şekildedir: önce, gövdesi τ ile dolu, τ'nun yapısını yukarıda belirtildiği gibi temsil eden doldurulmuş bir ağaç oluşturulur; sonra ağacın her bir boğumundaki (o boğumu dolduran) öğe bir sayıyla eşleştirilir. ρ ve μ ile eşleştirilen boğumlar sırasıyla n ve m ile eşleştirilmişse, $n + m$, $(\rho + \mu)$ ile eşleştirilir vb.

Aritmetik dilinin bölümsüz tamdeyimleri, τ ve ρ terim olmak kaydıyla, '$\tau < \rho$' ya da '$\tau = \rho$' biçimindeki ifadelerdir. *Tamdeyimler*, bölümsüz olsun olmasın, şöyle belirlenir:

- Her bölümsüz tamdeyim bir tamdeyimdir.
- Φ ve ψ tamdeyimse, şunlar da tamdeyimdir: $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$, $\backsim\phi$, (her bir n için) $(\exists x_n)Q$ ve $(\forall x_n)Q$.
- Yukarıda belirtilenlere uygun olmayan hiçbirşey tamdeyim değildir.

Terimler için olduğu gibi, tamdeyimler için de benzersiz okunabilirlik özelliği geçerlidir.

Bir x_n değişkeni, $(\forall x_n)$ ya da $(\exists x_n)$ ile başlayan bir alttamdeyimde geçiyorsa, ve ancak böyleyse, bir tamdeyimde *bağlı* olarak geçer. Bağlı olmayan geçişler *bağsızdır*. Bağsız değişken içermeyen tamdeyim, *cümledir*. Φ bir tamdeyim, τ de bir terim olduğunda, $\phi(x_n / \tau)$, ϕ'den, x_n'nin ϕ'deki her bağsız geçişi yerine τ konularak elde edilen tamdeyim olsun. Hangi değişkenin içerildiğine ilişkin bir karışıklık söz konusu olmadığında, bu tamdeyimi bazen basitçe $\phi(\tau)$ şeklinde göstereceğiz.

Standart modelde doğruluğu tanımlarken şunu yapacağız: ilk olarak bölümsüz cümleler için doğruluk koşullarını koyacağız, sonra karmaşık cümlelerin doğruluk koşullarının daha yalın cümlelerin doğruluk koşullarınca nasıl belirlendiğini göreceğiz. Standart modelin özgün bir niteliği şudur ki konuşma evreninin her bir üyesi bir rakamla adlandırılır. Bu özgün nitelik, verilecek olan doğruluk tanımını yalınlaştıracaktır, çünkü doğruluğu artık gerçekleme üzerinden değil, dolaysızca tanımlayabileceğiz.

- $\tau < \rho$ standart modelde, Den (τ) < Den (ρ) ise, ve ancak böyleyse, doğrudur.
- $\tau = \rho$ standart modelde, Den (τ) = Den (ρ) ise, ve ancak böyleyse, doğrudur.

- $(\phi \vee \psi)$ standart modelde, ϕ ve ψ'den biri ya da her ikisi standart modelde doğruysa, ve ancak böyleyse, doğrudur.
- $(\phi \wedge \psi)$ standart modelde, hem ϕ hem de ψ standart modelde doğruysa, ve ancak böyleyse, doğrudur.
- $(\phi \rightarrow \psi)$ standart modelde, ya ϕ standart modelde doğru değil, ya da ψ standart modelde doğruysa, ve ancak böyleyse, doğrudur.
- $(\phi \leftrightarrow \psi)$ standart modelde, ya hem ϕ hem de ψ, ya da ne ϕ ne de ψ standart modelde doğruysa, ve ancak böyleyse, doğrudur.
- $\backsim\phi$ standart modelde, ϕ standart modelde doğru değilse, ve ancak böyleyse, doğrudur.
- $(\exists x_n)Q$ standart modelde, en az bir k için $\phi(x_n / [k])$ standart modelde doğruysa, ve ancak böyleyse, doğrudur.
- $(\forall x_n)Q$ standart modelde, her k için $\phi(x_n / [k])$ standart modelde doğruysa, ve ancak böyleyse, doğrudur.
- ϕ standart modelde, $\backsim\phi$ standart modelde doğruysa, ve ancak böyleyse, yanlıştır.

Doğru aritmetik, standart modelde doğru olan cümlelerin kümesidir. Göreceğiz ki doğru aritmetik için bırakalım bir karar verme yöntemini, bir ispat yöntemi dahi yoktur. Ancak şundan da yoksun değiliz:

- **Önerme**: Niceleyicisiz doğru cümleler için bir karar verme yöntemi vardır.[3]
- **İspat**: İlk olarak doğru bölümsüz cümleler için bir karar verme yöntemi olduğunu kaydedelim: $\tau < \rho$, Den $(\tau) <$ Den (ρ) ise, ve ancak böyleyse, doğrudur; ve $\tau = \rho$, Den $(\tau) =$ Den (ρ) ise, ve ancak böyleyse, doğrudur. Niceleyicisiz bir cümlenin doğruluğu, cümlenin bölümsüz oluşturucularının doğru oluşu ya da olmayışına bağlı olarak, önermeler dizgesi yasaları bakımından belirlenir. Niceleyicisiz θ cümlesinin doğruluğunu ya da yanlışlığını belirlemenin bir yolu şudur: önce θ'nın yapı ağacı kurulur, sonra da yapraklardan gövdeye doğru, boğumları dolduran her bir öğeyle bir doğruluk değeri, yani doğru ya da yanlış değeri, eşleştirilir. Örneğin, $(\phi \vee \psi)$ öğesiyle dolu olan boğum, hemen altında yer alan iki boğumun biri ya da her ikisi 'doğru' ile doldurulmuşsa, ve ancak böyleyse, 'doğru' ile doldurulur. ■

Bağlı niceleyicileri şöyle tanımlıyoruz: ϕ tamdeyimi ve τ terimi için $(\exists x_n < \tau)Q$ ifadesi, $(\exists x_n)(x_n < \tau \wedge \phi)$ ifadesinin kısaltmasıdır. Aynı şekilde, $(\forall x_n < \tau)Q$ de, $(\forall x_n)(x_n < \tau \rightarrow \phi)$'nin kısaltmasıdır. *Bağlı tamdeyimler* de şöyle belirlenir:

- Her bölümsüz tamdeyim bağlıdır.

[3] Aksi söylenmedikçe, 'doğru' aritmetik cümleyle, standart modelde doğru olan aritmetik cümleyi anlatıyor olacağız.

- ϕ ve ψ bağlı tamdeyimse, şunlar da öyledir: $(\phi \vee \psi)$, $(\phi \wedge \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$, $\backsim\phi$, ve her bir n ve τ için geçerli olmak üzere $(\exists x_n < \tau)Q$ ve $(\forall x_n < \tau)Q$.

- Başka hiçbirşey tamdeyim değildir.

'$\rho < s\tau$' ifadesini kısaltmak için '$\rho \leq \tau$' ifadesini kullanıyoruz. Öyleyse, ϕ bağlı tamdeyim olduğunda, her bir n ve τ için $(\exists x_n < \tau)Q$ ve $(\forall x_n < \tau)Q$ de tamdeyim olacaktır.

Bağlı bir *küme*, bağlı bir tamdeyimin genişletimidir; başka deyişle, bağlı bir küme, ϕ gibi bağlı bir tamdeyim için $\{x: \phi([x])\}$ biçiminde bir kümedir. Aynı şey bağlı bağıntılar için de geçerlidir.

Çift, *birinci* ve *ikinci* fonksiyonları bağlıdır; x, kod sayısı y olan kümenin bir üyesi olduğunda x'le y arasında gerçekleşen bağıntı da öyledir ('$x \in y$' yazımını bu bağıntıyı da gösterecek şekilde kullanarak biraz genişletmiş olacağız). Dizilerin kod sayıları kümesi bağlı bir kümedir; ve yine, x, i uzunluğunda ya da daha uzun bir dizinin kodu olduğunda x ve i'yi bu dizinin (ki '$(x)_i$' şeklinde gösterilir) i-inci üyesine götüren, x, i'den kısa bir diziyi kodluyorsa x ve i'yi tanımsıza götüren kısmi fonksiyon da bağlıdır.

- **Önerme**: Doğru bağlı cümleler kümesi için bir karar verme yöntemi vardır.

- **İspat**: Dışarıdan içeriye doğru giderek, $(\forall x < \tau)\psi$ biçimindeki her alttamdeyimi, $k < \text{Den}(\tau)$ olmak kaydıyla, tüm $\psi([k])$'ların birletimiyle[4] değiştirelim. $(\exists xn < \tau)\psi$ ifadesini de yine $k < \text{Den}(\tau)$ için tüm $\psi([k])$'ların ayırtımıyla değiştirelim. Bunu, tüm bağlı niceleyiciler elenene kadar sürdürelim, sonra da geriye kalan niceleyicisiz bağsız cümlenin doğruluğunu denetleyelim. ■

  - **Ek sonuç**: Her bir bağlı küme için bir karar verme yöntemi vardır.

  - **İspat**: S bağlı bir kümeyse, $\phi(x)$ gibi bağlı bir tamdeyimin genişletimidir. n'nin S'de yer alıp almadığını anlamak için $\phi([n])$'nin doğru olup olmadığını denetleyelim. ■

Her bağlı küme karar verilebilirdir, ancak her karar verilebilir küme bağlı değildir. Bunu görmek için Cantor'un köşegenli çıkarımının bir örneğini kullanacağız. Bağlı bir tamdeyimi sözdizimi yapısından seçebiliriz, öyleyse tek bağsız değişkeni 'x' olan tüm bağlı tamdeyimleri dökümlemek mümkündür. Bu da bize tüm bağlı kümelerin dökümünü vermiş olacaktır. Şimdi, C = {n: n, dökümdeki n-inci bağlı kümenin üyesi değildir} olsun. C karar verilebilirdir. n'nin C'de yer alıp almadığını denetlemek için tek yapılacak olan dökümdeki n-inci tamdeyimi yazmak ve n'nin onu gerçekleyip gerçeklemediğine bakmaktır. Ne var ki C bağlı değildir. Çünkü öyle olsaydı, C = dökümdeki k-ıncı küme eşitliğini sağlayan k gibi bir sayı olurdu. O zaman da şunlarla karşı karşıya kalırdık:

$$k \in \text{dökümdeki k-ıncı küme}$$

$$k \in C \text{ ise (k'nın seçilme yolu dolayısıyla), ve ancak böyleyse}$$

[4] Bunun pürüzsüzce işlemesi için şu uylaşımı benimsiyoruz: $\{\psi\}$'nin birletimi yalnızca ψ, boş tamdeyim kümesinin birletimi ise '0 = 0'dır. Çok öğeli (yani boş kümeden farklı) bir tamdeyim kümesinin birletimi alınırken, birletilen öğelerin alınma sırası ve biraraya getirilme biçimi istenildiği gibi belirlenebilir. Benzer şekilde, $\{\psi\}$'nin ayırtımı ψ, boş tamdeyim kümesininkiyse '$\backsim 0 = 0$'dır.

k ∉ dökümdeki k sıra sayılı küme ise (C'nin tanımı dolayısıyla), ve ancak böyleyse

Karar verilebilir her küme için karar veren yazılımlar üretecek bir yazılım sağlama çabalarına karşı aynı çıkarımı kullanabiliriz. Öyle ki ya üretilen yazılımların bir bölümü (küme üyeliğiyle ilgili bazı soruları yanıtsız bırakacakları için) karar yöntemi olmayacak, ya da kendileri için hiçbir karar yöntemi üretilmemiş karar verilebilir kümeler olacaktır.

Yine de, etkilice sayılabilir olan tüm kümelerin dökümünü veren yazılımların dökümünü verecek bir yazılım sağlayabiliriz. S gibi herhangi bir karar verilebilir küme için, bu dökümde, biri S'yi diğeri S'nin tümleyenini sayan iki yazılım yer alacaktır. Bu iki yazılım, birlikte, S için karar veren bir yazılım sağlamış olur. Ne var ki bunu biliyor olmak, bize, karar verme yöntemlerini dökümleyecek bir yöntem kazandırmaz, çünkü elimizde S'nin sayımını yapan yazılımla onun tümleyeninin sayımını yapan yazılımı eşleştirecek bir algoritma yoktur.

Yine, hesaplanabilir bütüncül fonksiyonları hesaplayan yazılımların dökümünü veren hiçbir yazılım olmadığını gösterebiliriz. Bunu yapmaya kalkışan her yazılım ya bütüncül olmayan bir fonksiyona uygun bir yazılımı da dökümleyecek, ya da hesaplanabilir bir bütüncül fonksiyonu dışarıda bırakacaktır. Bunu *reductio ad absurdum*la görmek için diyelim ki böyle bir yazılımımız var. Şimdi *f* gibi şöyle bir bütüncül fonksiyon tanımlayalım:

$$f\ (n) = \ 1 \ + \ n - inci\ yazılımın\ n\ girdisine\ verdiği\ çıktı$$

f hesaplanabilir bir bütüncül fonksiyon olduğundan, dökümde *f*'yi hesaplayan bir yazılım bulunacaktır; diyelim ki bu, k-ıncı yazılımdır. Öyleyse,

k-ıncı yazılımın k girdisine verdiği çıktı

= (k'nın seçilme yolu dolayısıyla)

= 1 + k-ıncı yazılımın k girdisine verdiği çıktı (*f*'nin tanımlanma yolu dolayısıyla)

Böylece, elde edebileceklerimizin en iyisi, hesaplanabilir kısmi fonksiyonları dökümleyen yazılımları dökümleyecek bir yazılımdır.

Σ-tamdeyimleri, bağlı bir tamdeyime, bir varlık niceleyicileri bütününün önden eklenmesiyle elde edilen tamdeyimlerdir.

- **Önerme**: Doğru Σ-cümleleri için bir ispat yöntemi vardır.
- **İspat**: Örnek Σ-cümlesi, 'nin bağlı geçtiği, olsun. İspat yöntemi şöyledir: doğru bir cümle elde edene dek, ϕ'deki bağsız değişkenler yerine farklı farklı sıralı rakam k'lıları konur. ■

Bir Σ yöntemini ispatlamak için tüm yapmamız gereken bir tanık göstermektir. Bir Σ cümlesini çürütmek içinse sonsuz sayıda mümkün tanığı ortadan kaldırmamız gerekir ki bunun için bir algoritma

bulunmamasında şaşılacak bir şey yoktur. İleride de göreceğimiz gibi, doğru Σ tümceleri kümesi etkilice sayılabilir olmakla birlikte karar verilebilir değildir.

Bir Σ doğal sayılar kümesi bir Σ tamdeyiminin genişletimidir; başka deyişle, , eşitliğini karşılayan ϕ gibi bir Σ tamdeyimi varsa, ve ancak böyleyse, Σ'dır. Σ bağıntıları da benzer şekilde tanımlanır.

- **Ek sonuç**: Her Σ kümesi için bir ispat yöntemi vardır.
- **İspat**: S'nin ϕ (x) Σ-tamdeyimini gerçekleyen sayılar kümesi olduğunu kabul edelim. S'yi saymak için doğru Σ-cümleleri sayılarak başlanır, sonra da doğru Σ-cümleleri dökümüne ϕ ([n]) her eklendiğinde S'nin dökümüne de n eklenir. ■
- **Önerme**: S bir Σ-kümesiyse, ϕ (x, y) biçiminde öyle bir tamdeyim vardır ki S = {x: (∃y) ϕ ([x], y). Başka deyişle, bir Σ-kümesi tanımlamak için, bir varlık niceleyicileri kütlesine ihtiyacımız yoktur. Tek bir varlık niceleyicisi yeterli olacaktır.
- **İspat**: S bir Σ-kümesiyse, ψ (x, y_1, y_2,…,y_m) biçiminde, şunu karşılayan bir bağlı tamdeyim vardır: S = {x: (∃y_1) (∃y_2) … (∃y_m) ψ([x], y_1, y_2,…,y_m)}. O halde S = {x: (∃z) (∃y_1 < z) (∃y_2 < z) … (∃y_m < z) ψ ([x], y_1, y_2,…,y_m)}; '∃z' ifadesinden sonra gelen şey bir bağlı tamdeyimdir. ■

Bir varlık niceleyicileri kütlesiyle değil de bir bağlı tamdeyimin izlediği bir tümel niceleyiciler kütlesiyle başlarsak, elde edilen, bir Π-tamdeyimi olacaktır. Tümel niceleyiciler kütlesi yerine tek bir niceleyici konulabilir. Bir Π-tamdeyiminin genişletimi bir Π-kümesi olur. Böylece Π-kümeleri, Σ-kümelerinin tümleyenleridir. Hem Σ hem de Π olan bir kümenin Δ olduğu söylenir.

Şu kafa karıştırıcı terim kullanımının yerleşmiş olduğunu görüyoruz: deniliyor ki, bir küme ya da bağıntı, Δ ise, ve ancak böyleyse, *yinelemelidir*. Aynı şekilde, bir küme ya da bağıntı, Σ ise, ve ancak böyleyse, yinelemeyle sayılabilirdir. Buraya kadar iyi, ama bir Σ kısmi fonksiyonu için *kısmi yinelemeli fonksiyon* denildiğinde şu iki nedenden dolayı kafa karışıklığı doğar: birincisi, kullanılan ifade, yani 'yinelemeli kısmi fonksiyon' değil de 'kısmi yinelemeli fonksiyon' ifadesi, bir bütüncül yinelemeli fonksiyonun bir kısmının söz konusu olduğunu ima ediyor; oysa, ileride de göreceğimiz gibi, bütüncül yinelemeli fonksiyonlara genişletilemeyen kısmi yinelemeli fonksiyonlar da vardır. İkincisi, bu ifade bize diyor ki, (tek değişkenli) bir kısmi fonksiyon bir tür iki-yerli bağıntı olmakla birlikte, bir kısmi yinelemeli fonksiyon bir yinelemeli ikili bağıntı değil, yalnızca yinelemeyle sayılabilir bir bağıntıdır. Bu nedenle, kafa karışıklığı ihtimali bulunduğunda, 'yinelemeli' ve 'yinelemeyle sayılabilir' yerine sırasıyla 'Δ' ve 'Σ' ifadelerini kullanmaya çalışacağım. (Bir yinelemeli bütüncül fonksiyon her halükarda yinelemeli olduğu için, tam fonksiyonlar söz konusu olduğunda bu karışıklığa yer olmayacaktır.)

Bir bağlı kümenin genişletimi her durumda Σ'dır, çünkü bağlı tamdeyimin önüne etkisiz bir varlık niceleyicisi tutturabiliriz. Bir bağlı tamdeyimin değillemesi de bağlı olacağından buradan şu sonuç çıkar: bağlı tamdeyimin genişletimi her durumda Δ'dır.

İki Σ-kümesinin birleşimi de kesişimi de yine Σ'dır. R bir Σ bağıntısıysa, her k için, şunların hepsi Σ-kümesidir: {x: $(\exists y)Rx\}$, $\{x\colon (\exists y < k)Rxy\}$, ve $\{x\colon (\forall y < k)Rxy\}$.

Her Σ bütüncül fonksiyonu Δ olduğu gibi, Δ önalanlı her Σ kısmi fonksiyonu da Δ'dır. Bir küme, hususi fonksiyonu Δ ise, ve ancak böyleyse, Δ'dır.

❖ **Önerme**: Şunlar, S doğal sayılar kümesi için denktir:

S, Σ'dır.

S, bir Σ kısmi fonksiyonunun önalanıdır.

Önalanı S'yi kapsayan *f* gibi öyle bir Σ kısmi fonksiyonu vardır ki ise, ve ancak böyleyse, *f*(n) = 1.

R gibi öyle bir Δ bağıntısı vardır ki $S = \{x: (\exists y)R([x], y)\}$.

S ya boş kümedir ya da bir Σ bütüncül fonksiyonunun ardalanıdır.

S, önalanı doğal sayıların bir başlangıç dilimi olan bir Σ kısmi fonksiyonunun ardalanıdır.

S, bir Σ kısmi fonksiyonunun ardalanıdır.

S ya sonludur ya da birebir eşleşmeli bir Σ bütüncül fonksiyonunun ardalanıdır.

❖ **İspat**: Verilecek ispatlar, 'Σ'lar yerine 'etkilice sayılabilir' konularak elde edilecek önermelerin ispatlarıyla benzerdir. Burada tek ispatlamak istediğim, S'nin, Σ ise, ya sonlu ya da birebir eşleşen bir Σ bütüncül fonksiyonunun ardalanı olduğudur. S, Σ ise, ϕ bağlı tamdeyim olmak kaydıyla, S, $\{x: (\exists y)\phi([x], y)\}$ biçimindedir. S sonsuzsa, ardalanı S olan şöyle bir bütüncül fonksiyon tanımlayabiliriz:

$$f(0) = birinci\ (\phi(birinci(z), ikinci(z))'li\ en\ k\text{üçük}\ z)$$

$$f(1) = birinci\ (\phi(birinci(z), ikinci(z))\ ve\ birinci(z)'nin\ i \leq n\ için\ t\text{üm}\ f(i)'lerden\ farklı\ olduğu\ en\ k\text{üçük}\ z)$$

Başka deyişle,

$$\text{f(n)}=birinci\left(\phi\left(birinci(z), ikinci(z)\right) \wedge (\forall w < z)\left(ya\ \neg\phi\left(birinci(w), ikinci(w)\right) ya\ da\ (\exists i < n) f(i) = birinci(w)\right) için\ en\ k\text{üçük}\ z\right).$$

Başkaca söylersek, şu koşulları karışılayan, *m* uzunluğunda sonlu bir *s* dizisi varsa, ve ancak böyleyse, $f(x) = y$ olur:

$$(\forall n < m)(\exists z)\Big(\phi\big(birinci(z), ikinci(z)\big)$$
$$\wedge\ (\forall w < z)\Big(\neg\phi\big(birinci(w), ikinci(w)\big)\ \vee (\exists i < n)(s)_i = \text{birinci(w)}\Big)$$
$$\wedge\ \text{birinci(z)} = (s)_n\Big).$$

$$x < m\ \wedge\ (s)_x = \text{y}.$$

Tüm bunlar bir Σ-tamdeyimi olarak kodlanabilir. Yalnız yapılacaklar burada göze çarpandan biraz daha fazladır, çünkü 'birinci' ve 'ikinci' ifadeleri ve bir sonlu dizinin bileşenleri için kullanılan yazım öğeleri, aritmetik diline değil, sayılar hakkında gayriresmi olarak konuşurken kullandığımız doğal dile (bu durumda Türkçe'ye) aittir. 'y=birinci(x)', 'y=ikinci(x)', 'x, y uzunluğunda bir diziyi kodlar' ve 'x, y'inci bileşeni z olan bir diziyi kodlar' bağıntılarının hepsi artimetik dilinin bağlı tamdeyimleri olarak yazılabilir. Yani,

$$(\exists s)(\exists m \leq s)\Big(s, m\ uzunluğunda\ bir\ diziyi\ kodlar$$
$$\wedge\ (\forall n < m)\Big(\exists z$$
$$\leq s\Big((\exists u \leq z)(\exists v \leq z)\Big(\big(u = birinci(z) \wedge v = birinci(z)\big) \wedge \phi(u, v)\Big)$$
$$\wedge\ (\forall w$$
$$< z)\Big((\exists u \leq x)(\exists v \leq w)\big(u = birinci(w) \wedge v = ikinci(w) \wedge \neg\phi(u, v)\big)$$
$$\vee\ (\exists i < n)(\exists t \leq s)(s, i\ sıra\ sayılı\ \text{üyesi}\ t \wedge t$$
$$= birinci(w)\ olan\ diziyi\ kodlar)\Big)\Big)$$
$$\wedge\ (\exists u \leq z)(u$$
$$= birinci(z) \wedge s, n\ sıra\ sayılı\ \text{üyesi}\ u\ olan\ sonlu\ bir\ diziyi\ kodlar)\Big) \wedge x$$
$$< m \wedge s, x\ sıra\ sayılı\ \text{üyesi}\ y\ olan\ sonlu\ bir\ diziyi\ kodlar\Big)$$

ise, ve ancak böyleyse, $f(x) = y$.

Bunu ilkel yazımda yeniden yazdığımızda ve (bağlı değişkenlerin birbirlerine geçmemesine dikkat ederek) niceleyicileri öne çekmek için kullandığımız kuralları uyguladığımızda, gerçekten de bir Σ-tamdeyimi elde ederiz.■

Burada tanık olduğumuz şey aslında yinelemeli tanımları açık tanımlara dönüştürmek için Frege ve Gödel'ce kullanılmış genel bir yöntemin bir örneğidir. Şimdi bu yöntemi biraz daha soyutça betimleyelim. Tek değişkenli fonksiyonların yinelemeli tanımlarının bilindik iki biçimi vardır. (Çok değişkenliler için çok büyük bir değişiklik olmaz. Ek değişkenler aynı sürece katılırlar.) Bu iki biçimden daha yaygın olanı, fonksiyonun değerinin tamamen hemen bir önceki değerce belirlendiği durumlarda kullanılır. Böylece tanım şu biçimi alır:

$$f(0) = k$$

$$f(n+1) = g(n+1, f(n))$$

ki burada k bir sayı, g ise halihazırda bilinen bir (bütüncül) fonksiyondur. Bunu açık bir tanıma dönüştürmek için şu koşulu getiririz:

$$f(x) = y =_{tnm}$$

$$(\exists s\ dizisi)(\exists m \leq s)\big(m, s'ninuzunluğudur\ \wedge\ (s)_0 = k\ \wedge \big(n+1 < m'yi\ sağlayan\ her\ n\ için\ (s)_{n+1} = g(n+1, (s)_n)\big) \wedge x < m\ \wedge\ (s)_x = y\big).$$

Biraz daha açıkça:

$$(\exists s)(\exists m \leq s)(s, m\ uzunluğunda\ bir\ diziyi\ kodlar \wedge s, 0'ıncı\ üyesi\ [k]\ olan\ bir\ diziyi\ kodlar \wedge (\forall n < m)((n+[1]) < m \rightarrow (\exists w \leq s)(\exists z \leq s)(s, (n+[1])sıra\ sayılı\ üyesi\ w\ olan\ bir\ diziyi\ kodlar \wedge s, n\ sıra\ sayılı\ üyesi\ z\ olan\ bir\ diziyi\ kodlar \wedge g\big((n+[1], z) = w\big) \wedge (s, x\ sıra\ sayılı\ üyesi\ y\ olan\ bir\ diziyi\ kodlar).$$

Bütün bunları, ‘g((n+[1], z) = w)’ ifadesi yerine bir Σ-tamdeyimi koyduğumuzda, bir Σ-tamdeyimi olarak yazabiliriz.

Öteki biçim, ki yukarıda tanık olduk, fonksiyonun bir değerinin kendinden önceki bütün değerlere dayanıyor olduğu durumlarda kullanılır. f fonksiyonu için $f \restriction n$, $\langle f(0), f(1), \dots, f(n-1)\rangle$ dizisinin kod sayısı olsun. n uzunluğunda ya da daha uzun bir dizinin kod sayısı olan s için $s \restriction n$, $\langle (s)_0, (s)_1, \dots, (s)_{n-1}\rangle$ dizisinin kod sayısı olsun. (Aynı sembolü iki farklı amaç için kullandığımın farkındayım, ama bu bizim için sorun olmayacak.) Yinelemeli tanımımız şu biçimi alacaktır:

$$f(n) = h(f \restriction n)$$

Burada h halihazırda bilinmektedir. Bunun açık tanım hali şöyledir:

$$f(x) = y =_{tnm}$$

$$(\exists s\ sonlu\ dizisi)(\exists m \leq s)(m, s'nin\ uzunluğudur\ \wedge (\forall n < m)(s)_n = h(s \restriction n) \wedge\ x < m\ \wedge\ (s)_x = y).$$

$$h, \Sigma\ ise, f\ de\ \Sigma' dır.$$

Yine burada bir kümenin Σ olduğunu iddia ederken o kümeyi, hem formel hem de formel olmayan aritmetiğe ait simgeler içeren bir tür melez Σ-tamdeyimiyle tanımlayabilmeme dayanıyorum. Şimdi bu tür iddiaları meşru kılacak genel bir yöntemin betimlemesini vereyim. Bu yöntem yalnızca, bağlı

tümel niceleyicilerde geçen '→'ler dışında, '→' ya da '↔' içermeyen tamdeyimlere uygulanabiliyor, ancak bu bir genellik kaybına yol açmıyor. Şimdi, S kümesi, aritmetik diline, birli bir bütüncül yinelemeli fonksiyonu imlemek için kullanılagelen *f* gibi ek bir birli fonksiyon işareti katılarak elde edilmiş dildeki bir Σ-tamdeyimiyle tanımlanıyor olsun. Aynı yöntem, birden çok fonksiyon işareti kullanılarak tanımlanan kümeler ve bağıntılarla birlikte, birden çok işleniği olan fonksiyon işaretleri için de iş görecektir. Yazımı fazla karmaşıklaştırmamak için dikkatimizi tek başına duran birli fonksiyonlar örneğine vermekle yetineceğiz.

İzleyeceğimiz strateji basit: '*f*' ifadesini barındıran her bölümsüz tamdeyimi ya Σ tanımı ya da Π tanımı yoluyla uygun biçimde çözer, sonra da niceleyecileri en öne doğru çekeriz. Ne var ki bu işin ayrıntıları epey karmaşıktır.

Bir yinelemeli bütüncül fonksiyon hem Σ hem de Π'dir. Elimizde şu olsun: ψ ve θ bağlı olmak kaydıyla, $(\forall z)\theta(x,y,z)$ ise, ve ancak böyleyse, $(\exists z)\psi(x,y,z)$ ise, ve ancak böyleyse, $f(x)=y$. S'yi tanımlayan ve *f*'yi barındıran bir Σ-tamdeyimini yine S'yi tanımlayan ancak içinde f bir kez daha az geçen bir Σ-tamdeyimine dönüştürmek için bir yöntem tanımlayacağız. Bu yöntemi tekrar tekrar uygulayarak, S'yi tanımlayan ve aritmetik diline ait olan bir Σ-tamdeyimi elde edeceğiz.

Verili durumda *f* ya bir bölümsüz tamdeyimde, ya bağlı bir varlık niceleyicisi kapsamında ya da bir tümel niceleyici kapsamında geçer. Yöntemi betimlemeyi karmaşık kılan şey bu imkanlar çokluğu ve *f* için kimi zaman Σ kimi zaman da Π tanımını kullanıyor oluşumuzdur.

Bu yöntem iki aşamalıdır. Birinci aşamada, *f* verili olarak $\chi^{\upsilon}/_{f(\tau)}$ biçiminde bir bölümsüz tamdeyimin içinde yer alıyorsa iki seçeneğimiz vardır: bölümsüz tamdeyim, çift sayıda değilleme işaretinin etki alanındaysa tamdeyim $(\exists v)(\exists z)(\psi(\tau,\upsilon,z)\wedge\chi)$ ile, tek sayıda değilleme işaretinin etki alanındaysa $(\forall v)(\forall z)(\theta(\tau,\upsilon,z)\rightarrow\chi)$ ile değiştirilir. Verili olarak *f*, $(\exists v<\rho\,^{\upsilon}/_{f(\tau)})\chi$ biçiminde geçiyorsa, yine: bu geçiş çift sayıda değilleme işaretinin etki alanındaysa yerine $(\exists v)(\exists z)(\chi(\tau,\upsilon,z)\wedge(\exists u<\rho)\chi)$ konur, yok tek sayıda değilleme işaretinin etki alanındaysa yerine $(\forall u)(\forall z)(\theta(\tau,\upsilon,z)\rightarrow(\exists u<\rho)\chi)$ konur. Son olarak (!), 'nin verili geçişi $(\forall u<\rho\,^{\upsilon}/_{f(\tau)})\chi$ biçiminde olabilir. Bu durumda da, değillemenin çift ya da tek oluşuna göre bölümsüz tamdeyim yerine $(\exists v)(\exists z)(\psi(\tau,v,z)\wedge(\forall u<\rho)\chi)$ ya da $(\forall v)(\forall z)(\theta(\tau,v,z)\rightarrow(\forall u<\rho)\chi)$ konur.

Birinci aşama bize yeni bir bağsız değişken getirdi. İkincisiyse değilleme işaretini öne çekmenin bilindik yöntemlerini uygular. Bu yöntemler şunlardır[5]:

$\big((\exists v)\mu\vee\upsilon\big)$ yerine $(\exists v)(\mu\vee\upsilon)$ konur.

$(\upsilon\vee(\exists v)\mu)$ yerine $(\exists v)(\upsilon\vee\mu)$ konur.

[5] Burada v'nin υ içinde bağsız olmadığını varsayıyoruz; ayrıca, karışıklık olmaması için bağlı değişkenleri gerektiği gibi değiştiriyoruz.

$\big((\forall v)\mu \vee v\big)$ yerine $(\forall v)(\mu \vee v)$ konur.

$(v \vee (\forall v)\mu)$ yerine $(\forall v)(v \vee \mu)$ konur.

$\big((\exists v)\mu \wedge v\big)$ yerine $(\exists v)(\mu \wedge v)$ konur.

$(v \wedge (\exists v)\mu)$ yerine $(\exists v)(v \wedge \mu)$ konur.

$\big((\forall v)\mu \wedge v\big)$ yerine $(\forall v)(\mu \wedge v)$konur.

$(v \wedge (\forall v)\mu)$ yerine $(\forall v)(v \wedge \mu)$ konur.

$(\exists u < \tau)(\exists v)\mu$ yerine $(\exists v)(\exists u < \tau)\mu$ konur.

$(\exists u < \tau)(\forall v)\mu$ yerine $(\forall t)(\exists u < \tau)(\forall v < t)\mu$ konur.

$(\forall u < \tau)(\exists v)\mu$ yerine $(\exists t)(\forall u < \tau)(\exists v < t)\mu$ konur.

$(\forall u < \tau)(\forall v)\mu$ yerine $(\forall v)(\forall u < \tau)\mu$ konur.

$\neg(\exists v)\mu$ yerine $(\forall v)\neg\mu$ konur.

$\neg(\forall v)\mu$ yerine $(\exists v)\neg\mu$ konur.

En sonunda S'yi tanımlayan bir Σ-tamdeyimi elde edilir.

Şimdi, ispatları vermeksizin, Σ kümeleri ve bağıntıları için, daha önce etkilice sayılabilir kümeler ve bağıntılar için verdiğimiz özelliklere doğrudan benzer olan, iki yapı özelliği görelim:

- **Önerme (Σ kümeleri için indirgeme ilkesi)**: A ve B gibi her Σ kümesi için, şu özellikleri karşılayan C ve D gibi Σ kümeleri vardır: $C \subseteq A, D \subseteq B, C \cap D = \emptyset, ve\ C \cup D = A \cup B.$
- **Önerme (Σ bağıntıları için tekbiçimlileştirme ilkesi)**: R gibi her bir Σ bağıntısı için, şu özellikleri karşılayan f gibi bir Σ kısmi fonksiyonu vardır: $Den(f) = \{x: (\exists y) < x, y > \in R\}, ve\ x \in Dom(f)\ olan\ her\ x\ için < x, f(x) > \in R.$