`ModuleWithProviders` Migration

What does this schematic do?

Some Angular libraries, such as `@angular/router` and `@ngrx/store`, implement APIs that return a type called `ModuleWithProviders` (typically via a method named `forRoot()`). This type represents an `NgModule` along with additional providers. Angular version 9 deprecates use of `ModuleWithProviders` without an explicitly generic type, where the generic type refers to the type of the `NgModule`.

This schematic will add a generic type to any `ModuleWithProviders` usages that are missing the generic. In the example below, the type of the `NgModule` is `SomeModule`, so the schematic changes the type to be `ModuleWithProviders<SomeModule>`.

Before

```
@NgModule({...})
export class MyModule {
  static forRoot(config: SomeConfig): ModuleWithProviders {
    return {
      ngModule: SomeModule,
      providers: [
        {provide: SomeConfig, useValue: config}
      ]
    };
  }
}
```

After

```
@NgModule({...})
export class MyModule {
  static forRoot(config: SomeConfig): ModuleWithProviders<SomeModule> {
    return {
      ngModule: SomeModule,
      providers: [
        {provide: SomeConfig, useValue: config }
      ]
    };
  }
}
```

In the rare case that the schematic can't determine the type of `ModuleWithProviders`, you may see the schematic print a TODO comment to update the code manually.

Why is this migration necessary?

`ModuleWithProviders` has had the generic type since Angular version 7, but it has been optional. This has compiled because the `metadata.json` files contained all the metadata. With Ivy, `metadata.json` files are no longer required, so the framework cannot assume that one with the necessary types has been provided. Instead, Ivy relies on the generic type for `ModuleWithProviders` to get the correct type information.

For this reason, Angular version 9 deprecates `ModuleWithProviders` without a generic type. A future version of Angular will remove the default generic type, making an explicit type required.

Should I add the generic type when I add new `ModuleWithProviders` types to my application?

Yes, any time your code references the `ModuleWithProviders` type, it should have a generic type that matches the actual `NgModule` that is returned (for example, `ModuleWithProviders<MyModule>`).

What should I do if the schematic prints a TODO comment?

The schematic will print a TODO comment in the event that it cannot detect the correct generic for the `ModuleWithProviders` type. In this case, you'll want to manually add the correct generic to `ModuleWithProviders`. It should match the type of whichever `NgModule` is returned in the `ModuleWithProviders` object.

What does this mean for libraries?

Libraries should add the generic type to any usages of the `ModuleWithProviders` type.

What about applications using non-migrated libraries?

The Angular compatibility compiler (`ngcc`) should automatically transform any non-migrated libraries to generate the proper code.